SCHOOL OF ECONOMICS & MANAGEMENT LISBON

MESTRADO EM

DECISÃO ECONÓMICA E EMPRESARIAL

TRABALHO FINAL DE MESTRADO

RELATÓRIO DE ESTÁGIO

ESCALONAMENTO DE PRODUÇÃO COM TEMPOS DE *SETUP* DEPENDENTES – APLICAÇÃO NA SAPEC AGRO

POR: VASCO MOREIRA DA FONSECA BRANQUINHO

ORIENTAÇÃO:

PROFª DOUTORA MARIA CÂNDIDA VERGUEIRO MONTEIRO CIDADE MOURÃO

ENGENHEIRO NUNO ALEXANDRE CABAÇO CHORINCAS

09-2013

Agradecimentos

Nem as palavras mais bem empregues, serviriam para expressar a especial gratidão que tenho à minha professora orientadora, Doutora Maria Cândia Mourão do Instituto Superior de Economia e Gestão, que não só tornou este estudo e oportunidade profissional possível, como também, me apoiou em todos os momentos decisivos com a maior paciência, transmitindo-me sabedoria, motivação e confiança, proporcionando-me uma verdadeira experiência aprendizagem. Foi claramente o maior privilégio poder ser seu orientando.

Estou também, muito grato ao meu co-orientador Engenheiro Nuno Chorincas do departamento de logística da SAPEC Agro por toda a boa vontade que sempre teve em me receber, mostrando-me a empresa e disponibilidade total no fornecimento de dados que tanto me ajudaram na execução de todo o TFM. Foi graças ao meu co-orientador e ao seu plano de estágio que presenciei os mais diversos cenários da fábrica, havendo sempre disponibilidade para uma reunião diária onde se discutiram vários aspetos e características de todo o ambiente ao qual o meu trabalho se destina. A sua confiança e paciência revelaram-se fundamentais para tornar o presente estudo possível.

Eu gostava de deixar um agradecimento muito especial ao meu grande amigo João Pinto, finalista de engenharia eletrotécnica no Instituto Superior Técnico por todas as aulas, testes e *know-how* que me transmitiu e tão bem instruiu na área da programação. Foi graças a ele que aprendi a linguagem orientada a objetos Java. A sua inspiração e metodologia de ensino revelaram-se fundamentais para, através do lema "a máquina tem sempre razão", resolver imensos exercícios, criando uma base sólida para que todo o trabalho de programação aqui empregue viesse a surgir.

Eu gostava de agradecer ao mestrado de Decisão Económica e Empresarial do ISEG por toda a aprendizagem que me transmitiram e por me disponibilizarem o acesso a toda a base bibliográfica. Gostava igualmente de agradecer a Doutora Margarida Vaz Pato por se disponibilizar em ajudar durante todo o TFM. Ao Jorge Barros também fica um agradecimento especial por me ter dado ótimas sugestões durante a escrita do presente estudo.

Eu agradeço à SAPEC Agro por toda a experiência que me proporcionaram. Estou muito grato aos responsáveis da fábrica Luís, André e Sr. Rosa por toda a paciência que tiveram na resposta às mais diversas questões que tive. Ao Doutor Pedro Rosa, Engenheiro Mário Gomes, Doutora Ana Primo e todos os trabalhadores da linha de formulação de produtos pós molháveis também fica um agradecimento por me terem encaminhado ao longo do estágio.

Eu gostava ainda de agradecer a toda a minha família especialmente aos meus pais por me terem ajudado financeiramente e mostrado a maior paciência em todos os momentos mais complicados e de mais *stress*.

Abstract

The production process in the plant protection industry, as in many other industries, faces very often problems on how to schedule tasks that need to be completed in order to respond to a given set of orders. This study is made for the largest Portuguese multinational company in the agro-chemicals business, SAPEC Agro. SAPEC Agro relies on a winning strategy on this business, a business where competition is made between supply chains instead of companies. To follow the company's growth and its continuous internationalization process, inserted in a highly competitive environment, this project aims to maximize the automation and optimize the production planning, which is now made by an Industrial Engineer.

This study's main goal is to develop an algorithm that will be able to find the optimal solution (under certain constraints) for the scheduling of tasks in the production process. That means determining the best possible sequence of tasks to each machine in order to minimize the total makespan. This type of problem is classic in the literature, and it is known as a Job-Shop Scheduling Problem –with Sequence Dependent Setup Times (JSP-SDST). The resolution of the JSP-SDST is difficult, as in terms of computational complexity it is considered an NP-Hard problem, thus being a challenge in the areas of Operations Research and Computer Science.

As the problem is applied in an industrial environment, the scheduling is useful if the algorithms respond in a reasonable amount of time, allowing the production managers to get real-time support when decisions need to be taken.

The chosen approach was first to define an admissible solution space (some constraints to the allocations were applied), and then to find the optimum through a Branch-and-Bound method which uses a Depth-First-Search as method of search in the solution space. Other search methods and heuristics, also based on Branch-and-Bound are applied as well, in order to meet the time complexity constraints. A graphical interface is developed allowing its use even by those unfamiliar with the complexity of the problem. Users need just to include the inputs (orders and quantities of each product) and the program generates a schedule for the input orders. This work's main goal is the development of a program that would be useful and would add value to the organization in study, the SAPEC Agro.

Key Words

Scheduling; makespan; job-shop scheduling problem with sequence dependent setup times (JSP-SDST); NP-hard; Branch-and-Bound; *depth-first search.*

Resumo

No processo produtivo de uma indústria de fitofarmacêuticos deparamo-nos com problemas no sequenciamento e afetação de tarefas necessárias a encomendas. Este estudo destina-se à fábrica da maior multinacional portuguesa no ramo dos agro-químicos. A SAPEC Agro aposta numa estratégia vencedora no negócio agrícola e, neste ramo, a competição não é entre empresas, mas sim entre as cadeias de abastecimento. Com o crescimento da empresa e a sua internacionalização, inserida num ambiente altamente competitivo, torna-se fundamental automatizar o processo de planeamento da produção, que até aos dias de hoje tem sido feito por um engenheiro industrial.

O presente estudo tem como objetivo encontrar um sequenciamento de tarefas (*scheduling*), ou seja, determinar uma afetação ótima das tarefas às máquinas, minimizando o tempo de execução total (*makespan*). Este tipo de problema é um clássico da literatura e é conhecido como um problema de *job-shop scheduling* com uma sequência dependente de tempos de *setup* (JSP-SDST). Um JSP-SDST apresenta, uma complexidade *NP-difícil*. Sendo um problema de difícil otimização, constituiu, assim, um desfio nas áreas da Investigação Operacional e Ciência da Computação.

A abordagem escolhida passou primeiramente por definir o espaço de soluções admissíveis (S.A.) e neste encontrar uma solução ótima (sob determinadas condições), através do algoritmo *Branch-and-Bound,* recorrendo-se particularmente ao tipo de pesquisa *Depth First Search* (B&B-DFS) no espaço das soluções admissíveis. Neste estudo são apresentados outros tipos de pesquisa, também baseados em B&B, por forma a avaliá-los. Foi desenvolvida uma interface gráfica centrada no utilizador para que este possa usufruir do presente estudo, sem ter de lidar com a complexidade envolvida.

A interface gráfica implementada é viável e, em articulação com os procedimentos de otimização escolhidos, constitui uma mais-valia para a organização, esperando-se que possa vir a ser instrumento de trabalho futuro.

Palavras- Chave

Escalonamento; makespan; problemas de escalonamento com sequências dependentes de tempos setup (JSP-SDST); NP-difícil; Branch-and-Bound; *pesquisa em profundidade.*

Índice

Lista de Tabelas

Tabela:

Lista de Figuras

Figura:

Lista de Ilustrações

Ilustração:

Lista de Terminologias e abreviaturas

N-	conjunto de nodos do grafo disjuntivo.
C-	conjunto de arcos conjuntivos.
D-	conjunto de arcos disjuntivos.
P-	conjunto dos produtos.
J-	conjunto de tarefas.
M-	conjunto das máquinas.
n-	número total de tarefas.

m- número total de máquinas.

O_{ij}- operação *i* da tarefa *j*.

T_{ij}- instante em que operação *i* da tarefa *j* começa a ser processada.

p_{ij} tempo de processamento da operação *i* da tarefa j.

r_j - data de disponibilidade da tarefa *j* (*ready time*)

$S_{\alpha\beta}$- tempo de *setup* necessário entre qualquer transição de operações da família α para a β numa mesma máquina.

$S_{I\alpha}$- tempo de *setup* inicial da operação pertencente à família α.

Siglas:

(JSP-SDST)	*job-shop scheduling problem with sequence dependent setup times*
(B&B-DFS)	*Branch-and-Bound – Depth-Fisrt Search*
(B&B-BFS)	*Branch-and-Bound – Breadth-Fisrt Search*
(B&B-Best-FS)	*Branch-and-Bound – Best-First Search*
(B&B-RS)	*Branch-and-Bound – Random Search*
(FIFO)	*first in first out*
(WP)	*wettable powder*

Algoritmos:

B&B-DFS-	*Branch-and-Bound – Depth-Fisrt Search*
B&B-Best-FS-	*Branch-and-Bound – Best-First Search*
B&B-RS-	*Branch-and-Bound – Random Search*

Capítulo 1 A Empresa

Neste capítulo faz-se uma breve introdução da Empresa e tenta-se, tanto quanto possível, enquadrar as várias exigências de uma fábrica com as de um problema de sequenciamento de tarefas.

1.1 Introdução

A SAPEC foi fundada em Portugal em 1926, na Herdade das Praias em Setúbal, por forma a beneficiar da existência de linhas férreas e acesso ao mar. Inicialmente, vocacionou-se para a nutrição vegetal, primeiro na produção e comercialização de adubos tradicionais na Península Ibérica e, atualmente, como referência no segmento de adubação especial através da Tradecorp, presente em cerca de 100 países. Nos anos 60 do século XX, a empresa inicia o negócio da proteção das culturas, onde é definido um portfólio crescente de produtos genéricos.

Nos anos 80 do século passado, a empresa passa por uma restruturação profunda, adotando, a forma de *holding* de investimentos presente em diferentes ramos de atividades, mas mantendo, como aposta do Grupo, o setor agrícola. Assim, nos anos 90, assume a liderança no mercado português, que hoje em dia partilha com a Selectis (uma outra empresa do Grupo), no que diz respeito aos produtos genéricos.

No início deste século surge uma nova opção estratégica: ser um Grupo de referência no mercado de produtos genéricos diferenciados, independente e internacional. Para isto, no ano 2000, apostou na defesa europeia das moléculas fitofarmacêuticas como fator crítico de sucesso e em 2009 obtém a primeira aprovação. Hoje, após um investimento de mais de 30 milhões de euros, conta com mais de 40 moléculas aprovadas, num processo de registo próprio que se mantém ativo.

Atualmente, o Grupo SAPEC (observe a Figura 1.1), que no final do século operava exclusivamente em Portugal, exporta cerca de 70% do volume de negócios na área da proteção de culturas. Para tal conta com a maior empresa do grupo, a SAPEC Agro.

Figura 1.1- Empresas do Grupo SAPEC.

A SAPEC Agro serve o mercado nacional e internacional, apostando, assim, numa estratégia vencedora no negócio agrícola. Neste ramo, a competição não é entre empresas, mas sim entre as cadeias de abastecimento. Com o crescimento da empresa e a sua internacionalização, inserida num ambiente altamente competitivo, torna-se fundamental automatizar o processo de planeamento da produção, que, até aos dias de hoje, tem sido feito por um engenheiro industrial.

A SAPEC Agro dispõe de duas divisões: a Divisão de Planeamento e Controlo Industrial e a Divisão Industrial de Investigação e Desenvolvimento como ilustrado na Figura 1.2.

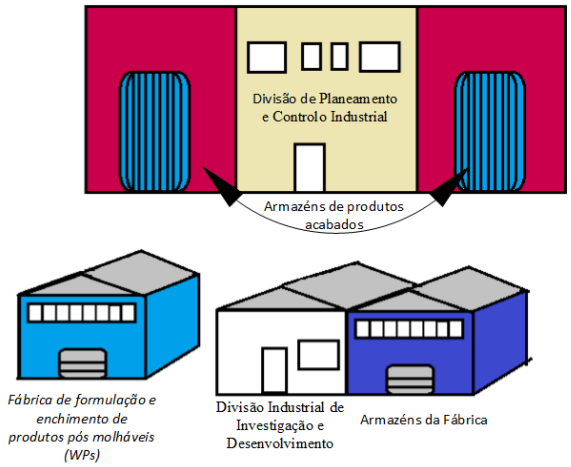


Figura 1.2- O planeamento da produção de produtos pós molháveis (*Wettable Powder*).

É no departamento de planeamento pertencente à Divisão de Controlo e Planeamento Industrial que, depois de definidos os planos de produção para a semana, se gerem as encomendas recebidas e se emitem as ordens de fabrico e enchimento dos produtos encomendados. Este departamento tem também a competência de gestão do *stock*, uma vez que para ser realizado um lote de um produto, numa dada data, é necessário que todos os componentes necessários estejam disponíveis atempadamente.

Estes componentes são descarregados tanto no porto de Lisboa, como no de Sines e, por vezes, existem atrasos que podem comprometer os prazos de entrega das encomendas.

No departamento de produção da Divisão Industrial de Investigação e Desenvolvimento recebem-se as ordens de fabrico, e estabelece-se uma estratégia de produção, de acordo com a disponibilidade dos componentes existentes em armazém e dos recursos nas datas específicas. Surge assim um problema de Sequenciamento de Tarefas (*Scheduling*), que constitui o objetivo de estudo neste trabalho. Segundo Pinedo (2012) "*Scheduling é o processo de tomada de decisão que é utilizado regularmente em muitas indústrias de transformação e de serviços. Trata-se de alocar os recursos a tarefas, num dado período de tempo e a sua finalidade é otimizar um ou mais objetivos"*.

As ordens de produção que se referem aos produtos do tipo pós molháveis (*Wettable Powder*, WP), fungicidas e inseticidas são, então, comunicadas aos responsáveis da fábrica que, por sua vez, decidem como alocar os trabalhadores às máquinas. Na fábrica existe um horário de laboração que pode ser alterado de acordo as exigências da produção. Normalmente, o horário de laboração é das 08:00 horas às 24:00 horas, funcionando em dois turnos. Em períodos de maior intensidade de produção, podem ser realizadas horas extraordinárias e, em casos extremos, criar um terceiro turno. Por outro lado, em períodos de menor intensidade de produção, a fábrica labora apenas num turno. Nos períodos de almoço e jantar, a produção é interrompida durante um tempo estritamente necessário para as refeições.

Na fábrica existem três tipos de produtos, os finais, os intermédios e os semi-acabados. Os produtos finais são embalados em pacotes de várias dimensões, ficando prontos para entregar aos clientes (distribuidores). Os semi-acabados são utilizados no fabrico de outros produtos.

As quantidades encomendadas têm uma forte componente sazonal. Nos períodos de verão a procura é bastante reduzida, mas com a chegada do inverno aumenta, atingindo o pico na primavera.

As encomendas resultam da execução de uma **tarefa** que se decompõem em uma ou mais **operações**: pré-mistura, moagem e mistura-final. Estas operações têm de ser processadas nas máquinas da fábrica por uma dada sequência. O problema de alocar tarefas às instalações de uma fábrica é bem reconhecido como um problema de difícil

resolução (O'Donovan et al., 1999). O tempo de processamento sequencial destas operações não depende só da velocidade da máquina que as executa, mas também do tempo de limpeza (*setup)* que poderá ser necessário entre quaisquer duas operações. Para além disto, o processamento de uma tarefa pode ter que ser atrasado, por falta de disponibilidade de máquinas, por existirem tarefas prioritárias, por tempos de *setup* morosos, por desempenho não esperado de certos trabalhadores ou por outro tipo de imprevistos.

Entre os imprevistos podem salientar-se avarias (temporárias) nas máquinas; atrasos no abastecimento dos componentes; operações que demorem mais tempo que o previsto devido a, por exemplo, enganos na quantidade de componentes de uma operação introduzidos nas máquinas; ou descarga de pós (inflamáveis) pelo facto de o reservatório não ficar bem colocado ou a tampa da máquina não ter sido bem fechada. Num ambiente destes, um plano de afetação das tarefas ajuda a manter a eficiência e o controlo dos recursos, conduzindo, consequente, a uma redução de custos do processo. A afetação apoia ainda atividades de planeamento como a decisão dos prazos de fornecimento dos componentes e entrega das encomendas aos clientes.

O fabrico de um produto, de acordo com a sua fórmula, implica o processamento sequencial de um determinado conjunto de operações. A duração das operações depende da máquina onde são realizadas, dado que estas têm velocidades de processamento distintas. Entre uma qualquer transição de operações numa mesma máquina poderá ser necessário efetuar uma limpeza. Este é um processo moroso que depende da diferença de tonalidades das cores dos dois produtos envolvidos.

1.2 Linha de fabrico

A linha de fabrico de produtos WP formula essencialmente dois tipos de pós molháveis: os fungicidas e os inseticidas. Quando formulados, os produtos (fungicidas ou inseticidas) são embalados na linha de enchimento.

É importante clarificar que cada lote produzido numa máquina, das sete existentes na secção de produção de WPs, tem 1 000 kg ou mais. As capacidades elétricas da fábrica não permitem o funcionamento em paralelo de todas as máquinas, contudo, seis destas podem processar continuamente operações (uma cada uma) durante

um, dois, ou os três turnos de um dia. Na verdade, estas inserem-se na linha com maior volume de produção, de entre um total de quatro linhas de fabrico da SAPEC Agro.

Um evento usual da linha de produção é o da indisponibilidade de máquinas, quer devido a manutenção preventiva quer por avaria. A indisponibilidade das máquinas que pode ser prevista (como no caso da manutenção) pode também ser incorporada aquando da afetação das tarefas. Contudo quando, no decorrer das tarefas diárias da linha de produção, uma máquina falha as operações que lhe estavam afetas não podem ser realizadas enquanto esta não se encontre reparada, uma vez que normalmente todas as instalações da fábrica estão a processar operações continuamente. Tal facto pode atrasar a conclusão de alguns produtos devido às precedências das operações que o envolvem (Hasan et al., 2010). Estes atrasos poderão, ainda, por sua vez, comprometer os prazos de entrega de encomendas, prejudicando, assim, a empresa.

1.2.1 O Big Bag

Na linha de formulação de WPs o *Big Bag* (Figura 1.3) é talvez o componente mais utilizado.

Figura 1.3- *Big Bag.*

É no *Big Bag* que se armazena um produto, durante todas as etapas produtivas, enquanto aguarda execução da próxima operação. Estes são "sacos" de grandes dimensões com uma abertura na parte superior e outra na inferior, e cordas para fechar e abrir as extremidades.

Os *Big Bag* assentam numa palete ou numa grade, de acordo com o tipo, e são transportados por hidráulicas manuais. Dado que a fábrica possui bastantes paletes, grades e *Big Bag*, estes não são considerados como recursos escassos nos modelos desenvolvidos.

1.2.2 As operações e as suas máquinas

Uma operação pode ser interrompida e retomada posteriormente sem qualquer penalidade. Esta situação pode ocorrer, por exemplo, a uma sexta-feira, sendo a operação interrompida durante o fim de semana, e retomada na segunda-feira ou, como constantemente acontece, no final do horário laboral. Contudo, a operação, depois de iniciada, só liberta a máquina quando pronta.

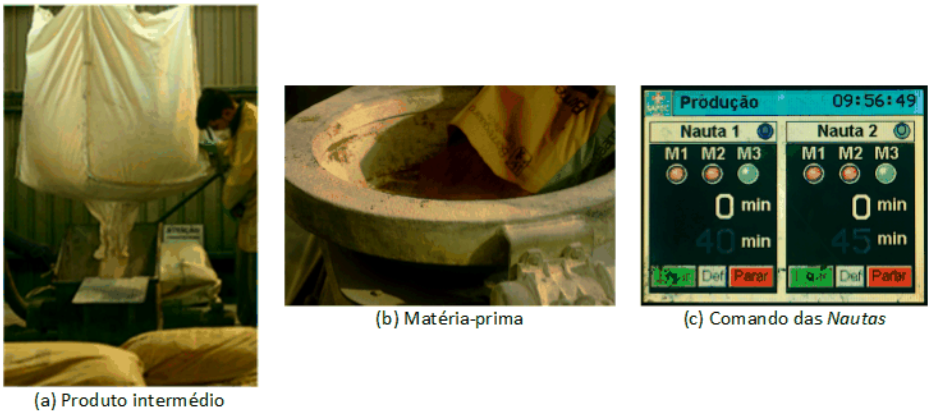
Para cada operação existem máquinas específicas que a podem processar e, **uma máquina só pode produzir uma operação de cada vez**.

Preparação

Inicialmente são atribuídas máquinas aos trabalhadores. Cabe a cada trabalhador ir buscar os vários componentes, ou sub-produtos, necessários à produção de uma operação e transportá-los para a respetiva máquina. Este tempo de preparação, válido em qualquer fase do produto nesta fábrica, é aproximadamente igual a 15 minutos. Assim, foram acrescentados 15 minutos a todas as fases de processamento das operações abaixo referidas, simplificando-se o problema.

Pré-mistura

A pré-mistura tem como objetivo a homogeneização e padronização dos componentes, que podem ser matérias-primas, produtos intermédios ou produtos semi-acabados, previamente preparados (observe a Figura 1.4).



(a) Produto intermédio (b) Matéria-prima (c) Comando das *Nautas*

Figura 1.4- Pré-mistura.

As máquinas que podem processar esta operação são a *Nauta 1*, a *Nauta 2* (*Nautas* verticais) e a *Misturadora 3* (*Nauta horizontal*) (observe a Figura 1.5). Cada uma destas máquinas está equipada com um *sem-fim* (Joseph, 1985) e (Nauta, 1979), que ao rodar mistura os componentes no interior do tanque, durante um determinado

tempo que é introduzido no comando das *Nautas* (observe Figura 1.4 (c)). O tempo de limpeza destas máquinas depende da sequência de operações resultante do escalonamento definido.

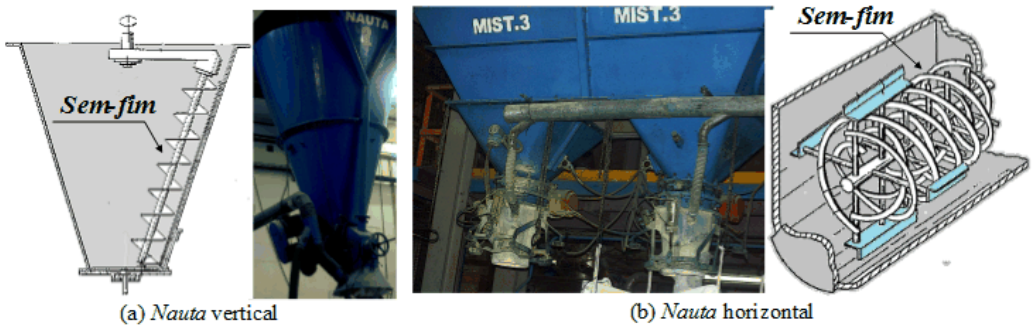


Figura 1.5- Máquinas misturadoras.

Por sua vez, a duração do tratamento da pré-mistura varia consoante o produto e máquina que a realiza. Esta operação surge frequentemente no fabrico dos produtos.

Moagem

A última operação de homogeneização do produto é a moagem dos grãos provenientes da pré-mistura. A operação de moagem é a mais morosa no processo produtivo, sendo processada em moinhos que podem ser de dois tipos: *Jet Mill e Alpine* (observe a Figura 1.6)

Figura 1.6- Máquinas de moagem.

O moinho *Alpine* tem no seu interior trinta e seis mangas para o transporte das matérias-primas ou produtos intermédios que vão ser moídos através de uma roda que os esmaga. Este tipo de moagem, embora bastante menos morosa, só se torna vantajosa para os produtos que se pretendem pouco moídos. Assim, quando se pretende uma moagem mais fina e controlável, a operação terá de passar várias vezes nesta máquina até se atingirem as características desejadas do produto. Tal, leva os responsáveis da

fábrica, a optarem, praticamente sempre quando deparados com tais produtos, pela moagem nos moinhos *Jet Mill*.

Nos moinhos *Jet Mill* as matérias-primas ou produtos intermédios são introduzidos e transportados através de um *sem-fim* para a câmara central do moinho. É nesta câmara que ocorre a pulverização, onde vários jatos de ar conduzem os componentes a uma velocidade super-sónica[1]. O ajuste da granulometria do produto é o resultado das colisões em alta velocidade entre as partículas do processo. O interior da câmara é concebido para permitir a recirculação de partículas de tamanho grande, aumentando a incidência e o efeito destas colisões. Assim, quando as partículas atingem o tamanho desejado são enviadas para o ponto de descarga central, tornando a classificação precisa, automática e controlável.

Mistura-final

A operação de finalização é bastante semelhante à de pré-mistura, quer em tempos de processamento e limpeza, quer nas máquinas que a podem realizar (*Nautas* e *Misturadora 3*). Nesta fase da produção podem ainda ser adicionados mais componentes (normalmente corantes), sejam eles matérias-primas, produtos intermédios ou semi-acabados, sendo necessário misturar o produto resultante. As máquinas disponíveis para realizar esta operação são as mesmas da pré-mistura ou seja, as *Nautas* e a Misturadora 3.

Na Tabela I é ilustrada a relação entre as operações e as máquinas que as executam.

Operações	Máquinas						
	Nauta 1	*Nauta 2*	*Misturadora 3*	*JetMill 1*	*JetMill 2*	*JetMill 3*	*Alpine*
Pré-mistura	X	X	X				
Moagem				X	X	X	X
Mistura Final	X	X	X				

Tabela I- Relação das operações com as máquinas da linha de produção de WPs.

1.2.3 Duração das operações

A duração de uma operação depende do produto a ser produzido e da máquina onde é processada, já que estas têm desempenhos diferentes. Uma fórmula estipula a

[1] A velocidade supersónica refere-se a qualquer velocidade acima da velocidade do som.

quantidade a processar, em quilogramas, de cada componente para a realização de cada uma das operações que constituem um lote de um produto (tarefa), bem como as máquinas que as podem executar.

Todas as operações têm tempos de execução determinísticos e são apresentados em minutos. Existem produtos que a SAPEC Agro já produz há muito tempo e, por isso, os tempos associados ao seu processamento foram estimados ao longo dos anos e fazem parte dos roteiros de produtos. Porém, existe um número considerável de tempos de processamento de operações que, por representarem produtos ainda em desenvolvimento ou recentemente estabelecidos, tiveram de ser estimados. Esta estimação decorreu da análise ao longo de três meses na secção da formulação de WPs, e foi validada pelos responsáveis envolvidos. Assim, foram registados todos os produtos estabelecidos até à data de entrega deste estudo, num ficheiro Excel com o nome "*static_info*" que será explicado mais à frente.

1.2.4 Limpezas das máquinas

Após serem utilizadas as máquinas poderá ser necessário proceder à sua limpeza. Os tempos de *setup*, quando existem, podem variar entre 1 e 16 horas de limpeza. Assim, estes, representam quer **custos consideráveis,** resultantes da alocação dos trabalhadores capacitados para proceder à limpeza das máquinas, quer **tempos ociosos** nas máquinas aquando desta operação. Deste modo, estes custos são assumidos neste modelo como tempos de limpeza (em minutos). Devido à sua estrutura, existem máquinas muito fáceis de limpar, como é o caso das misturadoras, outras com um processo de limpeza bastante moroso, como no caso do moinho *Alpine*.

O processo de limpeza mais complexo da fábrica é o da máquina *Alpine*. A limpeza desta máquina implica substituir as 36 mangas destinadas aos produtos anteriores pelas 36 mangas para os novos produtos. Este processo é muito moroso, levando dois turnos completos (16 horas) para estar concluído.

No que diz respeito às máquinas misturadoras e à *Alpine* há ainda que ter em atenção a sequência de cores dos produtos a processar, pois tal influencia o seu tempo de limpeza. Por exemplo, quando depois de se executar um produto azul-escuro (*Covicampo*) se pretende outro da mesma cor com uma tonalidade mais escura, não é necessário proceder à limpeza da máquina, visto não existir perigo de alterar a cor do produto seguinte. No caso extremo, se foi processado um produto azul-escuro

imediatamente antes de um branco, a máquina tem de ser perfeitamente limpa, o que demora cerca de quatro horas.

Considerando as máquinas misturadoras, quando a transição é feita entre operações de diferentes cores cuja tonalidade é sucessivamente mais escura; ou sucessivamente mais clara (excetuando-se os produtos brancos) não é considerado qualquer tempo na limpeza da máquina. Na situação inversa os tempos de limpeza são elevados, porque é necessário uma limpeza mais rigorosa da máquina. Assim, a **sequência de cores das operações** processadas numa máquina determina o seu tempo de limpeza.

Por política da empresa, os moinhos *Jet Mill* são dedicados a operações brancas ou, em raros casos, amarelas. Quer isto dizer que, se um destes produtos contiver algum tipo de corante ou cobre, este só será adicionado na mistura-final e, por isso, a limpeza destas máquinas não é necessária.

Assim, os tempos de *setup* dependem da sequência de cores das operações.

Capítulo 2 *Branch & Bound* para a minimização do *makespan*

Este capítulo divide-se em seis secções.

A **secção 2.1** começa com um enquadramento do problema, incluindo-se a pesquisa feita. Na **secção 2.2** formaliza-se o problema no, bem conhecido, grafo disjuntivo. Na **secção 2.3** é desenvolvido o algoritmo *branch-and-bound* (B&B) através dos três métodos de pesquisa aqui propostos: *depth-first search* (DFS), *best-first search* (Best-FS) e *random search* (RS). Na **secção 2.4** são apresentados os resultados computacionais dos diferentes algoritmos para pequenas instâncias. Na **secção 2.5** apresenta-se uma comparação dos resultados computacionais entre duas metodologias propostas com os resultados da produção realizada atualmente na secção de formulação de WPs da fábrica, nas primeiras oito semanas deste ano.

2.1 Introdução

Como referido no capítulo anterior, o fabrico de cada produto (tarefa) exige a realização de várias operações que têm de ser processadas por uma determinada ordem.

Assim, podemos definir este problema de sequenciamento como o problema de agendamento de operações (respeitando todas as precedências entre elas), num determinado número de máquinas não idênticas, considerando os **tempos de *setup*** (incluindo custos) e as **durações das operações**, com o objetivo de minimizar o tempo de execução total (***makespan*)**.

(Allahverdi et al., 2008) apresentam uma revisão de literatura dos problemas de escalonamento com tempos de *setup* ou custos, focando, essencialmente, os últimos vinte e cinco anos. Referem ainda que o estudo destes problemas começou a meio dos anos sessenta do século XX. O problema de *job shop scheduling*, com uma sequência de tempos *setup* dependentes (*JSP-SDST)* em apenas uma máquina, e com o objetivo de minimizar o *makespan* pode ser transformado num problema do caixeiro viajante, que é *NP-difícil* ((J. Park, 2012) e (Pinedo, 2012)). A complexidade aumenta à medida que são consideradas mais máquinas. "*O JSP-SDST é obviamente um problema NP-hard, uma vez que admite o problema de job shop como um caso particular que, por sua vez, também é NP-hard*" (Artigues et al., 2004). *Contudo, apesar das suas similaridades, poucos trabalhos foram desenvolvidos no sentido de o otimizar"*. Foram, no entanto, desenvolvidas algumas abordagens eficientes para o resolver ((Cheung and Zhou, 2001); (Choi and Choi, 2002); (Ballicu et al., 2002); e (Balas et al., 2008)). Segundo, (Pinedo, 2012), *"a minimização do makespan num job shop é um problema muito difícil de otimizar (...). Para obter soluções ótimas são exigidos métodos branch-and-bound* ". Assim, este estudo, por lidar com um problema de difícil otimização, constitui um desafio nas áreas da Investigação Operacional e Ciências da Computação.

A utilização do método *Branch-and-Bound* garante a minimização do *makespan*, contudo, para grandes instâncias do problema o seu desempenho não é o mais razoável. Ao longo dos anos foram propostos muitos métodos heurísticos por diversos investigadores com vista à resolução deste problema (como por exemplo, (Cheung and Zhou, 2001), (Choi and Choi, 2002) e (Artigues, 2003)). O objetivo deste trabalho é também o de propor uma heurística para o escalonamento dos produtos na secção de formulação de WP's.

2.2 A Minimização do makespan

2.2.1 Estratégia de resolução

A estratégia de resolução aqui empregue é a pesquisa *Branch & Bound*, com vista ao apuramento da melhor solução possível, minimizando o *makespan*. Um *makespan* mínimo geralmente garante uma boa utilização das máquinas.

A formalização do problema, que se apresenta de seguida, foi um passo essencial para a implementação do algoritmo. No entanto, esta formalização simplifica o caso real da SAPEC Agro, uma vez que não admite **recirculação** (um produto apenas poderá visitar uma mesma máquina uma e só uma vez) e, como será explicado na secção 2.2.3, supõe que **cada operação só pode ser realizada numa máquina** (na secção de WPs as tarefas de mistura podem ser realizadas em três máquinas misturadoras e as de moagem em quatro máquinas de moagem).

2.2.2 Formalização

Seja, o conjunto $J = \{J_1, J_2, \dots, J_n\}$ de n **tarefas** (produtos), definido por sequências ordenadas de **operações** por forma a que cada tarefa $J_j (j = 1, \dots, n)$ se decomponha em m operações $O_{1j}, O_{2j}, \dots, O_{mj}$ a serem processadas num conjunto $M = \{m_1, m_2, \dots, m_m\}$ de m **máquinas**, por uma determinada ordem. O conjunto de todas as operações é dado por O e a sua dimensão é $\left[\sum_{j=1}^{n} \sum_{i=1}^{m} O_{ij}\right]$. Na secção de formulação de WPs existem máquinas em paralelo com diferentes velocidades. Assim, o **tempo de processamento** p_{ij} da operação $O_{ij} \in O$ depende da máquina $m_i \in M$ (por simplificação, considera-se que uma máquina pode ser representada apenas pelo índice $i = 1, \dots, m$), a processá-la. As durações das tarefas estão expressas em minutos. Para que um produto seja realizado é necessário que todos os componentes estabelecidos na fórmula se encontrem disponíveis numa dada data, $(readyTime), r_j$, denominada como **data de disponibilidade** da tarefa *j*.

Sendo este problema identificado como um problema de *job shop* com *m* máquinas, a sequência de operações tem de obedecer às relações de precedência. Uma operação sucessora só poderá ser executada quando a sua precedente foi realizada. Esta é denominada por **precedência do problema** e não pode ser violada. Em termos de

implementação, só quando uma operação precedente é executada, é que a operação sucessora é libertada para poder ser alocada.

No que diz respeito aos produtos, estes podem ser catalogados como produtos prioritários, sem prioridade ou lotes a corrigir (se não passaram no controlo de qualidade).

Quando uma operação O_{ij} é concluída numa máquina m_i e se pretende executar outra operação O_{il}, poderá existir um tempo de *setup*. Para o considerar, as *m* operações foram agrupadas em **famílias de cores** num conjunto $F = \{\alpha, \beta \dots, \gamma\}$. Operações da mesma família processadas sequencialmente têm associado um tempo de *setup* nulo. No entanto, se houver uma alteração de famílias de produtos, digamos da família α para a β numa mesma máquina então é exigido um tempo de limpeza (*setup*), $S_{\alpha\beta}$. Para este problema foram definidas quatro famílias de cores ($|F| = 4$): branca, w, amarela, y, verde, g e azul, b. Para além disto, poderá ser necessário um tempo de *setup* inicial $S_{I\alpha}$ se a operação pertencente à família α for a primeira a ser processada na respetiva máquina e assim for exigido.

Como referido, por vezes as máquinas avariam, no decorrer da atividade da fábrica. A indisponibilidade de uma máquina tanto pode ser conhecida antes de feito o escalonamento, como ser imprevisível. Uma máquina que se encontre indisponível não poderá executar operações por um período de tempo, até que o problema técnico seja resolvido. O estudo não abrange esta situação e sempre que este cenário (imprevisível) ocorrer deverá ser efetuado um novo escalonamento.

Dada uma qualquer sequência, é possível calcular, para cada tarefa $j \in J$, a sua data de conclusão, C_j. Neste trabalho, o critério de otimalidade usado é a minimização do tempo total de execução, $C_{max} = \max_j C_j$.

2.2.3 Grafo disjuntivo

A minimização do *makespan* num *job shop* pode ser representada por um grafo disjuntivo. Inicialmente proposta por (Roy, 1964), continua a ser a representação a que recorre a maioria dos autores (como, por exemplo, (Pinedo, 2012), (Ballicu et al., 2002), (Artigues et al., 2004) e (Zoghby et al., 2005)). Este grafo permite-nos representar as diferentes precedências e ter uma melhor perceção dos tempos *setup* nas respetivas máquinas.

Considere-se então, o grafo orientado G, com um conjunto de nodos $\boldsymbol{N}$ e dois conjuntos de arcos *C* e *D*. Os **nodos** correspondem a operações das tarefas em causa e a dois nodos fictícios, o de **origem** *I* e o de **destino** *F*. Do nodo fictício I partem *n* arcos conjuntivos que correspondem à primeira operação *i* de cada tarefa j (O_{ij}), com tempo igual ao tempo de *setup* inicial da respetiva operação. Por sua vez, ao nó F chegam *n* arcos conjuntivos que dizem respeito à ultima operação *m* de cada tarefa j (O_{mj}), com tempo igual ao tempo de processamento da respetiva operação na máquina. Com exceção dos nodos fictícios, os restantes são representados por $(i,j)\ \forall\ i \in M\ \wedge\ j \in J$. Deste modo, $|N| = \left[\sum_{j=1}^{n}\sum_{i=1}^{m} O_{ij}\right] + 2$.

Os **arcos conjuntivos de** $\boldsymbol{C}$ (orientados e a cheio), $[(i,j),(i+1,j)] \in C$ representam as restrições de precedência e entre o par de operações $O_{ij}, O_{(i+1)j} \in O$, a que se associa um tempo de processamento $p_{ij} > 0$. Entre duas operações de tarefas distintas $O_{ij}, O_{il} \in O : O_{ij} \in \alpha; O_{il} \in \beta$ que possam ser processadas na mesma máquina m_i poderá, como referido, existir um tempo de *setup* (custo) dado por $S_{\alpha\beta}$. A ordem pela qual as operações podem ser processadas numa mesma máquina é representada pelos chamados **arcos disjuntivos de** $\boldsymbol{D}$ (orientados e a tracejado), $[(i,j),(i,l)] \in D$ com tempo $p_{il} + S_{\alpha\beta}$.

Os arcos disjuntivos de *D* formam *m* ***cliques***, uma para cada máquina, de arcos duplos, com sentidos opostos entre si. Segundo Pinedo (2012) *"uma clique é um grafo em que dois quaisquer dos seus nodos estão ligados entre si. Neste caso, cada ligação entre dois nodos de uma clique representa um par de arcos disjuntivos"*. O **grafo** assim construído, denominado **disjuntivo**, é denotado por $G = (N, C, D)$.

Exemplo: Para melhor explicar o problema que surge na linha de fabrico, considere-se o seguinte exemplo, representado na Figura 2.1, em que se pretende encontrar um escalonamento admissível para uma pequena instância do problema. Neste, foram encomendados 3 produtos (tarefas) diferentes, J_1, J_2 e J_3. O processo de fabrico das tarefas envolve o processamento sequencial de operações com os respetivos tempos de processamento, como se ilustra na Tabela II:

Tarefa	Operação	Tempo de processamento
J_1	O_{11}, O_{21} e O_{31}	$p_{11}, p_{21} e\ p_{31}$
J_2	O_{22} e O_{32}	$p_{23}\ e\ p_{32}$
J_3	O_{13}, O_{23} e O_{33}	p_{13}, p_{23} e p_{33}

Tabela II- Dados referentes aos arcos conjuntivos.

As sequências de operações de uma tarefa são então representadas por arcos conjuntivos, entre nodos numerados por forma a indicar o par "operação/tarefa" e coloridos por forma a indicar a família a que pertencem.

Cada máquina pode realizar um conjunto de operações representada nos arcos disjuntivos. A importância dos custos incorridos em tempos *setup* ilustra-se na Tabela III e na Tabela IV referente aos tempos (em minutos) de *setup* $S_{\alpha\beta}$:

Máquina	Operação	Família de cor
Misturadora m_1	Pré-mistura: O_{11} (branca) e O_{13} (branca)	$w; w$
Moagem m_2	Moagem: O_{21} (amarela) , O_{22} (azul) e O_{23}(verde)	$y; b; g$
Misturadora m_3	Mistura final: O_{32} (azul), O_{31} (verde) e O_{33} (azul)	$b; g; b$

Tabela III- Dados referentes aos arcos disjuntivos.

		$\boldsymbol{\beta}$			
		w	*y*	*g*	*B*
$\boldsymbol{\alpha}$	*w*	-	-	-	-
	y	60	-	-	-
	g	180	-	-	-
	b	240	120	-	-

Tabela IV- Matriz de tempos *setup* (em minutos) para tansições entre famílias α e β.

Observe-se então a representação do grafo disjuntivo para o exemplo.

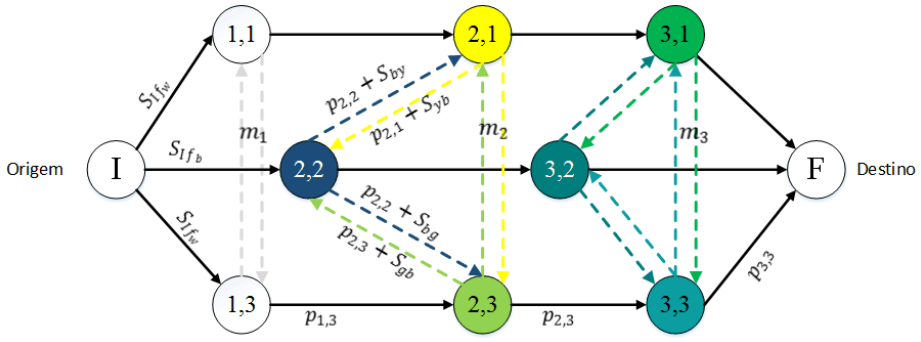


Figura 2.1- Grafo orientado para um problema de *JSP-SDST* para a minimização do *makespan*.

Um escalonamento admissível corresponde a uma *seleção* de um só arco disjuntivo entre cada par de arcos opostos por forma a que o respetivo grafo orientado seja acíclico. Para que tal aconteça, a *seleção* dos arcos disjuntivos de cada *clique* tem que ser acíclica. A ordem pela qual a seleção dos arcos é feita dentro de uma clique, determina a ordem pela qual as operações são processadas numa máquina e a existência ou não de tempos de *setup*. Note-se que a seleção dos arcos disjuntivos tem de originar uma rede acíclica, uma vez que se existisse um ciclo a sequência de operações nessa máquina seria não admissível, pois violava as restrições de precedência impostas nos arcos conjuntivos. Para ilustrar esta afirmação, considere-se a Figura 2.2. Este grafo diz respeito a três tarefas (j, l, k) a serem processadas em duas máquinas (h e i). Como se pode obeservar, existe um ciclo ($O_{ij} \rightarrow O_{il} \rightarrow O_{hl} \rightarrow O_{hj} \rightarrow O_{ij}$). Este ciclo é formado com 2 arcos conjuntivos e 2 disjuntivos, tornando a solução não admissível, uma vez que, O_{ij} é processada primeiro que O_{hj}, enquanto os arcos conjuntivos para a tarefa j impunham a ordem oposta.

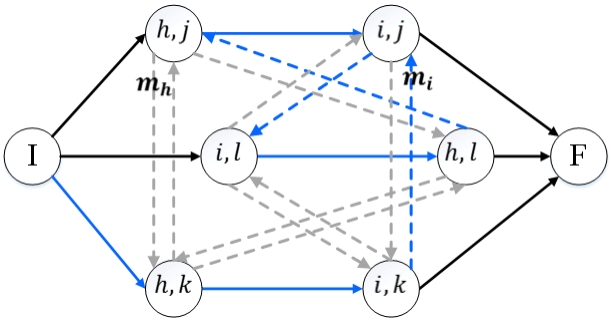


Figura 2.2- Um ciclo dentro de uma *clique*.

Assim, seja E o subconjunto de arcos disjuntivos selecionados e o subgrafo $G(E)$ definido pela união entre o conjunto de arcos conjuntivos e E. Então, E corresponde a um escalonamento admissível se e só se $G(E)$ não contiver circuitos (ciclos orientados). Assim, sempre que se verifica um escalonamento admissível, estamos perante uma **solução admissível (S.A.)**.

O *makespan* de um escalonamento admissível é determinado pelo caminho mais longo no subgrafo correspondente, $G(E)$, da origem I para o destino F, ou seja, é constituído pelo conjunto de seleções admissíveis dos arcos disjuntivos, que respeite as restrições impostas nos arcos conjuntivos, por forma, a que a primeira operação comece no momento zero (sem tempos *setup* iniciais) e a última seja concluída no instante de tempo correspondente ao *makespan*.

Uma representação de uma solução admissível referente ao grafo do exemplo anterior (Figura 2.1) considerando o conjunto de arestas disjuntivas E resultante das sequências de operações $\{(O_{11}, O_{13}); (O_{22}, O_{21}, O_{23}); O_{32}, O_{31}, O_{33}\}$ é representada abaixo no diagrama de Gantt (observe a Figura 2.3). Os tempos de *setup* estão representados a roxo, enquanto os blocos cinzento claros referem-se a tempos ociosos (*hidle*) da máquina.

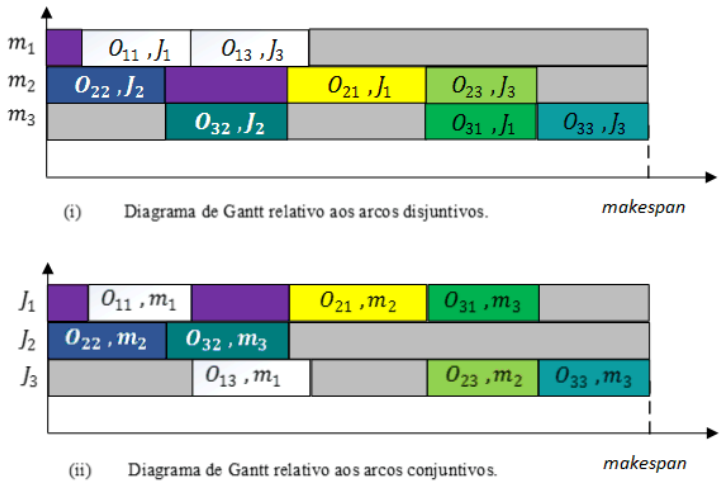


Figura 2.3- Diagramas de Gantt para uma solução admissível do *JSP-SDST*.

Em (i), na Figura 2.3, está representada a solução admissível para cada uma das máquinas. Existe um tempo de *setup* inicial em m_1, para processar a pré-mistura (O_{11}) da tarefa J_1. Por outro lado, em m_2 existe um tempo de *setup* entre duas operações pertencentes a famílias de cores diferentes: $O_{22} \rightarrow O_{21}$, dado por S_{by}. Quando m_1 e m_2 concluem as suas operações ficam *idle* até que a máquina do caminho crítico, m_3, conclua as suas operações. Em cada máquina (uma *clique*) foi feita uma seleção dos arcos disjuntivos acíclica, e assim, o diagrama de Gantt resultante para as tarefas não viola as restrições impostas pelos arcos conjuntivos (obeserve-se a Figura 2.3 em (ii)).

2.3 *Algoritmo Branch-and-Bound*

Os métodos *branch-and-bound* (B&B) aplicados a um problema de escalonamento têm uma estrutura especial. Nestes pode considerar-se o caso real da SAPEC Agro e, assim, os modelos admitem recirculação e permitem que uma operação possa ser realizada em mais que uma máquina. No entanto, na formalização apresentada do modelo esta questão não foi admitida por forma a simplificar a exposição.

2.3.1 Programação disjuntiva

O algoritmo branch-and-bound (B&B) é implementado através da programação disjuntiva, relacionada com a representação simplificada do grafo disjuntivo atrás referido.

Para ser apresentada a programação disjuntiva, considere-se a variável T_{ij} que representa o instante de tempo em que a operação O_{ij} se inicia. Seja, agora, N o conjunto de todas as operações $O_{ij} \in O$, e C o conjunto de todas as restrições de precedência (de uma tarefa *j*), $(O_{ij} \rightarrow O_{(i+1)j})$ que exigem que a tarefa *j* tenha de ser processada primeiro na máquina *i*, antes de ser processada na máquina (*i*+1). A seguinte formalização tem como função objetivo a minimização do *makespan* (1).

(1) $Min\ Z = C_{max}$

Sujeito a:

(2) $T_{ij} + p_{ij} \leq T_{kj} \qquad \forall\big((i,j),(k,j)\big) \in C$

(3) $C_{max} - T_{mj} \geq p_{mj} \qquad [(m,j),F] \in C$

(4) $\big(T_{ij} \geq T_{il} + p_{il} + S_{\alpha\beta}\big) \vee\ (T_{il} \geq T_{ij} + p_{ij} + S_{\beta\alpha})$
$\forall\big((i,j),(i,l)\big) \in D: O_{ij} \in \beta; O_{il} \in \alpha)$

(5) $T_{ij} \geq S_{I\alpha} \qquad \forall\big(I,(i,j)\big) \in C: O_{ij} \in \alpha$

(6) $T_{ij} \geq 0 \qquad \forall(i,j) \in N$

(7) $C_{max} \in \mathbb{N}$

Nesta formalização (2) asseguram que uma operação sucessora da tarefa *j* não pode começar, sem que, a operação precedente da mesma tarefa se encontre concluída. As condições em (3) definem C_{max} como o *makespan*. As restrições (4) são definidas por disjunção e ordenam operações de diferentes tarefas que são processadas numa mesma máquina. É devido a estas restrições que este tipo de formalização se denomina programação disjuntiva. Por fim, o tempo de *setup* inicial, para uma operação $O_{ij} \in O$ de uma família $\alpha \in F$ que seja a primeira a ser processada numa dada máquina, é definido em (5).

2.3.2 O algoritmo e o espaço de soluções

Os métodos *branch-and-bound* (B&B) aplicados a um problema de escalonamento permitem explorar o espaço de soluções admissíveis (S.A.). Para isso, foram seguidas determinadas terminologias que se explicam nesta secção. Assim, considere-se a **definição** proposta por (Schrage, 1972), que diz respeito a uma classe de escalonamento:

Definição- Escalonamento imediato (*active schedule*). *Um escalonamento admissível é dito de imediato se não puder ser alterado de nenhuma forma, permitindo que, uma operação possa estar concluída mais cedo sem que, pelo menos, uma outra seja adiada*.

O conjunto dos escalonamentos imediatos abrange todas as S.A. em que não é possível realizar, no diagrama de Gantt, um deslocamento global, ou local, de uma operação $O_{ij} \in O$ para a esquerda, sem que se aumente o valor do *makespan.* Numa afetação imediata sempre que uma operação se encontra disponível para ser alocada numa máquina, então, esta é processada tão cedo quanto possível, respeitando a ordem de precedências. Deste modo, as máquinas que processam essas operações não incorrem em períodos ociosos, exceção feita para casos em que não tenham operações para processar.

Num escalonamento imediato é pois impossível reduzir o *makespan*, a menos que se atrase o instante de início de uma operação quando existe pelo menos uma máquina habilitada a processá-la, o que, por sua vez, não é permitido, pois, nenhuma máquina pode ter um período ocioso sem que se justifique. É claro que existem muitos escalonamentos imediatos.

O B&B implementado segundo a programação disjuntiva e tendo em conta esta classe de escalonamentos apresenta uma caraterística: o **espaço de soluções** é constituído apenas por **soluções admissíveis (S.A.)** e, pelo menos, uma delas é a solução ótima. Assim, neste estudo apenas serão considerados os escalonamentos imediatos por forma a explorar, no espaço de S.A., aqueles que minimizam o *makespan* ($makespan^*$).

Segundo (Sprecher et al., 1995) *"para problemas de escalonamento com uma medida de desempenho regular, como a minimização do makespan, a solução ótima será sempre um conjunto de afetações imediatas".* O objetivo deste estudo é, como referido, o de encontrar uma solução que minimize o *makespan*. Note-se que o

makespan é uma medida de desempenho regular, pois é crescente com os tempos de conclusão das tarefas.

O B&B implementado é baseado no algoritmo proposto por (Pinedo, 2012). Este representa a base para o processo de B&B e é através deste que se geram todas as afetações imediatas, como será explicado nesta secção e se detalha no pseudocódigo do Anexo A.2.

Existem similaridades entre o algoritmo que se apresenta em seguida e o grafo disjuntivo atrás referido. Note-se que uma S.A. corresponde a uma *seleção* de um arco disjuntivo entre cada par de arcos opostos, por forma a que o respetivo grafo orientado seja acíclico. Para que tal aconteça a *seleção* dos arcos disjuntivos de cada *clique* tem também de ser acíclica.

Em cada **seleção** é então tomada uma direção nos arcos disjuntivos opostos. Com ligação a partir de I existe um conjunto omega $\Omega = \{\Omega_1, \dots, \Omega_k\}, k \leq m,$ de k cliques Ω_i, definido por todas as operações O_{ij} sem precedentes que cada máquina i está habilitada a processar. Por exemplo, na inicialização, em Ω_1 cada operação O_{1j} (pré-mistura) de uma família $\alpha \in F$ de cada tarefa j tem um instante de começo $T_{1j} \geq 0$ e existe uma máquina m_{1^*} habilitada a processar uma destas operações. Para uma melhor perceção observe-se novamente a Figura 2.1 (página 16).

Um nó V, do primeiro nível da árvore de B&B, corresponde a uma afetação parcial em que é selecionada uma das operações O_{1j}, ou seja, é escolhido um arco disjuntivo da clique $\Omega_1 = \{((1,1),(1,3)),((1,3),(1,1))\}$. Cada afetação parcial representa então uma seleção de um arco disjuntivo ($O_{ij} \in \Omega_i$ alocado a m_{i^*}) que identificará a ordem pela qual as operações serão realizadas (afetas a m_{i^*}), desde a origem I até esse dado nó V. Esta ordem determina o respetivo tempo total de *setup* da máquina m_{i^*}. Note-se que se um nó pai V resultou da alocação da operação $O_{1j} \in \Omega_1$ que, por sua vez, precede uma operação O_{2j}. Esta só estará pronta a começar num instante $T_{2j} \geq p_{1j} + T_{1j}$, por forma a não violar as restrições impostas pelos arcos conjuntivos. Deste modo, quando O_{1j} se encontra concluída, é então removida de Ω_1 e a operação O_{2j} é inserida em Ω_2. Assim, o número de ramificações possíveis no nó V (proveniente de Ω_1) é igual ao número de operações ainda por selecionar em Ω_2.

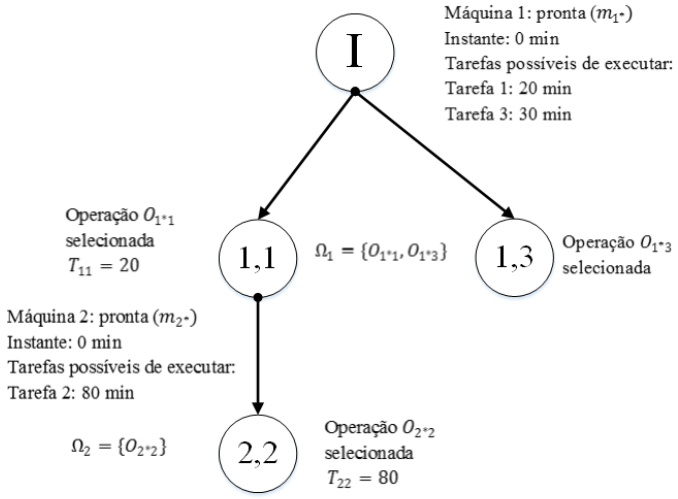


Figura 2.4- Árvore de ramificações para a abordagem *Branch-and-Bound*.

As ramificações são feitas máquina a máquina (*clique a clique*). Assim, escolhe-se uma máquina $m_{(i+1)^*}$ que seja a próxima a estar livre (pronta), num dado instante $T_{(i+1)^* j} \geq 0$. Neste instante, aloca-se, caso exista, uma operação $O_{(1+1)j} \in O: (i < m)$ em Ω_{i+1} pronta a começar na máquina $m_{(i+1)^*}$, tão cedo quanto possível ($T_{(i+1)^* j}$). Se não existir nenhuma operação nestas condições, então a máquina é colocada em espera (*idle*) e analisa-se a próxima máquina. Assim, identificam-se todas as possíveis alocações à máquina $m_{(i+1)^*}$, em número igual ao número de operações prontas a começar em Ω_{i+1}. Na Figura 2.4, por exemplo, considera-se apenas uma alocação possível no instante $T_{2^* j}$ para a máquina m_{2^*}. Cria-se um nó filho na árvore para esta operação O_{22} em Ω_2. O nó filho, V', corresponde, então, a uma afetação parcial que inclui, adicionalmente, a operação $O_{2^* j}$. O processo é repetido e quando $\Omega_i = \emptyset, \forall i \Rightarrow \Omega = \{\}$, atinge-se o último nível da árvore, $\sum_{j=1}^{n} \sum_{i=1}^{m} O_{ij}$, ou seja, determina-se um escalonamento imediato.

Neste instante, o algoritmo, dependendo do tipo de pesquisa considerada, parte para um outro nó ainda não explorado e repete o processo de ramificação até ser encontrado outro escalonamento imediato, ou até, a pesquisa ser cancelada. Seja E' o conjunto de arcos disjuntivos já selecionados, no nó recentemente criado, e $G(E')$ o grafo que contém todos os arcos conjuntivos e os de E'. Repare-se que E' é construído à medida que uma operação O_{ij}, por já estar concluída, é removida de Ω_i e inserida neste conjunto. Um dado nó da árvore de B&B, V' correspondente ao grafo $G(E')$, é

considerado um insucesso se a duração do caminho crítico nesse nó, $LB(V')$, é igual ou maior que o melhor *makespan* encontrado até ao momento, associado a um escalonamento imediato. Neste caso, a pesquisa nesse nó é cancelada, pois $LB(V')$ representa um minorante para o valor ótimo. O algoritmo analisa de seguida um nó ainda não visitado, e examina todos os nós filhos por ele gerados. O processo de ramificação termina quando não restarem quaisquer operações para alocar, ou seja, quando tiver sido criado um subgrafo $G(E')$ incluindo todos os nodos *N*.

Quando forem gerados todos os nós da árvore, os do último nível representam todos os escalonamentos imediatos. Assim, com este algoritmo pretende-se explorar o espaço de S.A.. Cada S.A., representa uma seleção de arcos disjuntivos, no grafo $G(E)$, por forma a que o respetivo grafo solução seja acíclico. A resolução do JSP-SDST, consistindo na minimização do *makespan*, corresponde a encontrar o mínimo dos caminhos mais longos nos respetivos grafos $G(E)$, sendo o *makespan*, o comprimento desse caminho.

Todos os procedimentos referidos no presente capítulo, até esta secção, foram implementados no pacote *BB_Colors* e são ilustrados no diagrama de classes com cor de fundo verde no Anexo A.3.

2.3.3 Limitações do B&B

Neste estudo escolheu-se gerar todas as S.A. de uma classe de escalonamentos, os escalonamentos imediatos. Assim, segundo as condições aqui seguidas, sempre que uma máquina estiver pronta a alocar uma operação, $O_{ij} \in O$, e, como esta não pode ser atrasada, a operação é afeta à máquina. Contudo, esta abordagem pode apresentar desvantagens.

A título exemplificativo, imagine-se um caso em que apenas falta uma operação $O_{ij} \in \alpha$ para ser alocada e a única máquina que está pronta a processar O_{ij}, nesse instante, é a máquina m_{i^*} que acabou de executar uma operação $O_{il} \in \beta$. Neste momento, o algoritmo identificando uma máquina disponível e habilitada a processar uma operação, aloca-a de imediato à máquina. A duração da operação na máquina *i*, é $d_{ij} = p_{ij} + S_{\beta\alpha} = 10 + 20$ originando um *makespan* igual a C_{max}. Considere agora que passados 5 minutos de a operação O_{ij} ser alocada à máquina m_{i^*}, uma outra máquina m_{k^*}, que estava a processar uma operação $O_{kh} \in \alpha$, fica livre (pronta) e, por

acaso, também podia ter processado a operação O_{ij}, com igual tempo de processamento, $p_{kj} = 10$ e sem tempo de *setup*. Sendo possível, o *makespan* seria então alterado para $C'_{max} = C_{max} - (p_{ij} + S_{\beta\alpha}) + p_{kj} = C_{max} - 30 + 10 < C_{max}$. Como as máquinas nunca ficam *idle*, não se considera possível esta solução, prejudicando o valor do *makespan* (por não ser necessário um tempo de *setup*). Esta é uma desvantagem do algoritmo.

Comprova-se assim que, uma solução correspondente a um escalonamento imediato poderá exigir um conjunto de tempos de limpeza que poderiam ser evitados, melhorando o *makespan*. Contudo, estes tempos de *setup*, também só pioram a solução se afetarem os instantes de começo das tarefas a ser processadas na máquina que está no caminho crítico. De qualquer modo, estas foram as hipóteses assumidas.

No B&B podem ser considerados, em geral, quatro tipos de pesquisa: *Depth-First Search* (DFS); *Best-First Search* (Best-FS); *Breadth-First Search* (BFS); Aleatória (RS- *Random Search*).

No DFS a pesquisa é feita em profundidade, fundamentalmente por razões de memória. Note-se que face a um varrimento em largura (BFS) a memória ocupada com o DFS é muito menor. Para ser encontrada uma solução admissível, no DFS, o número de nós corresponde à altura da árvore, uma vez que cada nó resulta de uma alocação de uma tarefa a uma máquina. No final, a solução admissível resulta do número total de tarefas alocadas (nós). Assim, a árvore do B&B cresce linearmente com o número de tarefas enquanto com o BFS a memória ocupada cresce exponencialmente com o número de tarefas, como ilustrado na Figura 2.5.

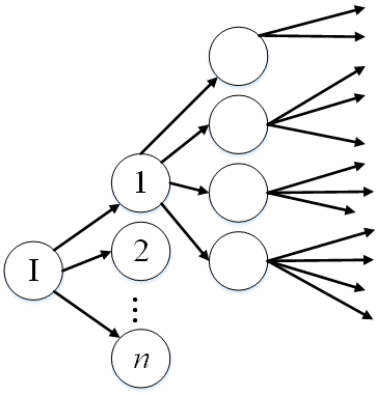


Figura 2.5- árvore de B&B com pesquisa BFS.

Na estratégia de pesquisa *Best-Fisrt Search* (Best-FS), em cada iteração aloca-se a tarefa cuja duração é a menor, considerando todas as tarefas que a máquina em causa pode processar. Este tipo de pesquisa, contudo, exige muito espaço de memória para guardar os nós do conjunto de tarefas sem tarefas precedentes na fila de prioridades. De facto, para grandes instâncias do problema, o número de nós a guardar pode exceder as capacidades de memória dos computadores, tornando-se impraticável.

Na estratégia de pesquisa aleatória (RS), a pesquisa é feita em profundidade, da mesma forma que em DFS. Contudo, como será explicado na secção 2.3.7, este tipo de pesquisa não explora o espaço total de S.A., quer por razões de memória, quer por razões de tempo de resposta. Para além disto, esta pesquisa apresenta uma característica: a flexibilidade na seleção dos nós da árvore, uma vez que esta é feita de forma aleatória.

Na Tabela V apresentam-se os tipos de procura em árvore referidos neste estudo e as suas características.

Tipo de Procura	Estrutura de Dados Utilizada	Complexidade Memória	Complexidade – Tempo (Pior Caso)
Depth First Search - DFS	Pilha	Linear com o número de tarefas	Exponencial com o número de tarefas
Breadth First Search - BFS	Fila Simples	Exponencial com o número de tarefas	Exponencial com o número de tarefas
Best First Search – Best- FS	Fila de Prioridade	Exponencial com o número de tarefas (mas menor que BFS)	Exponencial com o número de tarefas
Random Search RS	Lista de aleatoriedade	Limitada pelo número de iterações	Limitada pelo número de iterações

Tabela V- Tipos de procura em árvore e as suas características

2.3.4 Detalhes da Implementação

A linguagem de programação escolhida para implementar os algoritmos foi o Java. Esta adapta-se a uma implementação orientada a objetos. Cada **objeto** representa uma instância da própria classe. Em cada classe definem-se três tipos de membros: atributos, métodos e construtores. É nos **atributos** que se armazenam os dados para os objetos da classe. Por sua vez, os **métodos** estabelecem o que cada objeto executa e os **construtores** armazenam o código responsável por inicializar os atributos dos objetos. Na Figura 2.6 apresenta-se a classe produto, uma das classes definidas neste estudo.

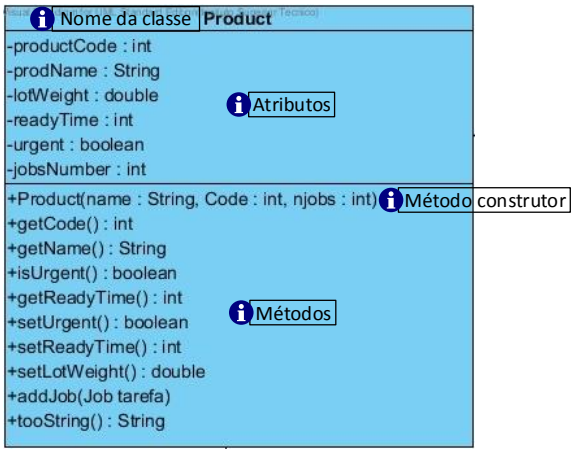


Figura 2.6- A classe produto

Numa linguagem orientada a objetos especificam-se quais as classes que serão instanciadas pelo *software*, de modo a que este execute os diferentes algoritmos. Uma forma simples de ilustrar a arquitetura implementada é através do diagrama de classes que deve representar uma visão abstrata do sistema, através das associações estabelecidas entre as diferentes classes. Esta abstração facilita a compreensão do sistema. É através do diagrama de classes (ver Anexo A.3) que se define o estado e o comportamento de cada classe.

A informação relativa aos dados do problema foi introduzida num ficheiro Excel com o nome "*static_info*". Esta base de dados é constituída por cinco folhas: tarefas, produtos, máquinas, encomendas e limpezas. A estratégia utilizada pretende possibilitar ao utilizador uma forma simples de alterar os dados, num ambiente bem conhecido como é o Excel (sem ter de conhecer o código), face a uma mudança que ocorra.

Por forma a modelar cada nó do B&B foi criada a classe *BBNode* (os métodos desta classe pertencem ao pacote *BB_Colors* e podem ser consultados no Anexo A.1). Esta tem como objetivo guardar a informação do problema no nó atual, ou seja, operações já realizadas, operações por realizar, hipóteses ainda por ramificar, fila de espera de máquinas ocupadas, solução temporária e custo respetivo. Na Figura 2.7 ilustra-se a inicialização do algoritmo implementado. Ao nó do nível zero, foi dado o nome de *MasterNode*.

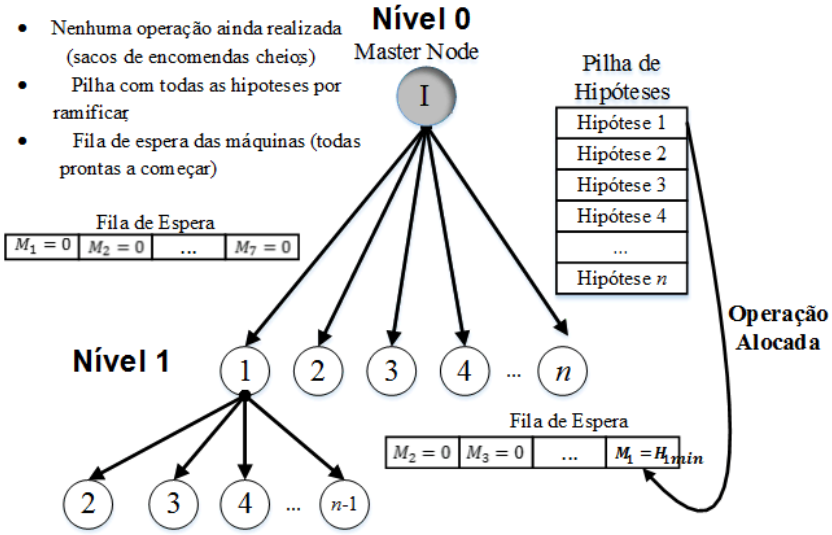


Figura 2.7- Implementação da metodologia *B&B-DFS*

Na pesquisa em profundidade (DFS), a análise é de tipo *LIFO* (o último a entrar é o primeiro a sair) e, assim, recorre-se a uma *pilha* para armazenar as possíveis ramificações (hipóteses) a partir de cada nó. Através das operações fundamentais de uma *pilha* é possível adicionar uma hipótese (*push*), aceder (*peek*) ou remover (*pop*) a hipótese que está no topo da pilha.

Por outro lado, em cada nó da árvore, é fundamental dispormos de informação dos instantes em que cada máquina estará pronta a alocar uma nova operação. Deste modo, foi utilizada uma fila de espera, representada numa *fila de prioridades*, em que cada elemento da fila diz respeito a uma máquina. A prioridade definida resulta do instante em que a máquina fica disponível para processar uma nova operação. A fila encontra-se ordenada de forma crescente de prioridades. Assim, a primeira posição da fila corresponde à próxima máquina a estar disponível para processar uma nova operação (máquina i^*). Esta é a única posição acedível através do método *peek*. Por outro lado, o último elemento contém a máquina que ficará disponível em último lugar.

Durante a pesquisa as operações realizadas são decrementadas dos *"sacos"* e adicionadas à lista de operações alocadas da respetiva máquina. Se uma operação $O_{ij} \in O$ realizada libertar uma operação sucessora $O_{(i+1)j} \in O$, então insere-se esta no respetivo saco no intante $T_{(i+1)j} \geq T_{ij} + p_{ij}$. Assim, as listas definidas podem ser de dois tipos *LinkedList* e *ArrayList*.

A *LinkedList* permite, num tempo constante, inserir ou remover elementos através da utilização dos métodos de interface *ListIterator* (para percorrer a lista). No entanto, para se obter uma dada posição da lista, o tempo exigido é proporcional ao seu tamanho.

Numa *ArrayList,* o acesso de leitura a uma posição qualquer da lista é feito em tempo constante, através dos métodos $get()$ e $set()$, independentemente do tamanho da lista. No entanto, perde-se eficiência na adição ou remoção de elementos, por ser necessário deslocar todos os elementos ao longo da lista para realizar tais operações. Para além disto, se for necessário acrescentar um elemento à lista, e for excedida a capacidade da matriz subjacente, é criada uma nova lista com o dobro do tamanho e copia-se a velha lista para a nova.

Estes aspetos foram tidos em conta na definição de todas as estruturas de dados que suportam os algoritmos com o objetivo de otimizar o seu desempenho. As estruturas de dados implementadas fazem parte do pacote *Data_Structurs* com cor de fundo amarela e são ilustradas no diagrama de classes no Anexo A.3.

2.3.5 Pesquisa em profundidade- DFS

O B&B-DFS exigiu a implementação da classe ***Simulator_BBDFS***. É nesta classe que é feita, com recurso ao método público da classe ***BBNode***- genChild(), a pesquisa/simulação de todas as afetações imediatas. Cada uma (inclusive com tarefas por realizar) corresponde a uma instância da classe *BBNode*. O simulador utiliza uma pilha de instâncias *BBNode*, com a finalidade de guardar os nós "pai" quando inicia a pesquisa de um filho. Assim, a informação dos nós pendentes fica sempre salvaguardada. O método público *genChild*() gera os próximos filhos do nó em causa e, analisados todos os filhos, retorna *null*.

O ciclo termina quando a pilha de nós se encontrar vazia o que significa que foi encontrada a melhor solução, de acordo com os pressupostos assumidos. Na Figura 2.8 apresenta-se o fluxograma desta pesquisa.

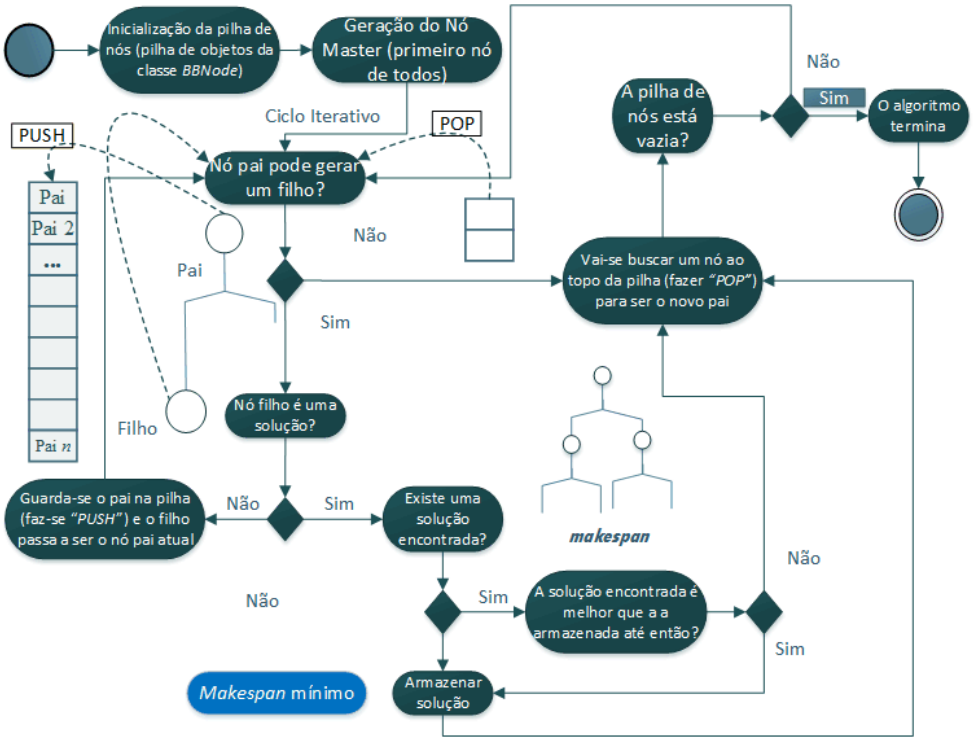


Figura 2.8- Fluxograma da pesquisa *Branch-and-Bound – DFS*

2.3.6 Pesquisa de tipo Best-FS

A estratégia de *best-first search* (Best-FS) é um método de pesquisa *greedy*, uma vez que, em cada iteração, é feita a melhor escolha possível, de entre o total de tarefas que a máquina pode processar. Foi então definida a classe ***SimultorBestFS*** para pesquisa das melhores alocações possíveis, com recurso ao método público da classe ***BBNode***-genChild(). Cada alocação (inclusive as com tarefas por realizar) corresponde a uma instância da classe *BBNode*. O simulador utiliza uma **fila de prioridade** de instâncias *BBNode*, posicionando na cabeça da fila o objeto *BBNode* com o menor *makespan*. Assim, quando a fila se encontrar vazia a solução obtida é um ótimo global.

2.3.7 Pesquisa aleatória- RS

Como referido para grandes instâncias, como as resultantes de um planeamento semanal, a pesquisa Best-FS pode exceder a capacidade de memória do computador (1 a 4 GB), tornando-se impraticável. Por outro lado, no B&B-DFS, a pesquisa vai sendo feita em profundidade. Na inicialização do algoritmo (nível zero) o conjunto das hipóteses é constituído por todas as tarefas que podem começar. Destas, é selecionada uma e, uma vez encontrada uma solução, é feita uma pesquisa local ao longo de todas as

sub-ramificações provenientes do primeiro nó selecionado. Uma vez explorado, parte-se para a próxima hipótese e repete-se o processo. Deste modo, para grandes instâncias (número de tarefas superior a 40), a altura da árvore torna a pesquisa exaustiva e o tempo de resposta muito moroso.

Por forma a desenvolver um algoritmo que pudesse, em tempo útil, encontrar uma solução admissível e sem exceder as capacidades de memória, foi desenvolvida a pesquisa aleatória. Assim, suponha que pretendemos estudar, de entre a população de soluções admissíveis (constituída por todos os escalonamentos imediatos) uma, observada que minimize o *makespan*. Estas são obtidas através da seleção dos arcos disjuntivos do grafo $G(E')$, por forma a que todos tenham uma direção e o resultante grafo seja acíclico. Uma vez que para grandes instâncias esta população, embora finita, é demasiadamente grande, pretende-se recolher uma amostra aleatória simples sem reposição de nós da árvore, que conduzam a soluções admissíveis, para podermos identificar qual a que tem o *makespan* mínimo.

Em suma, com o algoritmo *Branch-and-Bound Random Search* (B&B-RS), a pesquisa segue a seleção aleatória de um nó $V' \in \Omega$, que desenvolve uma solução admissível. Assim que encontrada uma solução admissível, é selecionado, de forma aleatória, um outro nó $V'' \in \Omega$ e repete-se o processo. Cada nó do conjunto de tarefas sem precedentes Ω, tem uma probabilidade de ser selecionado dada por $\left(P_{O_{ij}} = \frac{1}{|\Omega_i|}\right)(O_{ij} \in O \subset \Omega_i)$. É utilizada uma função do Java que gera estas seleções de forma aleatória. Esta abordagem permite percorrer o espaço de S.A. (população) de forma flexível (evitando-se a exaustiva pesquisa local), sendo geradas várias S.A. (amostra), numa amostra de 2 000 000 iterações, e recorrendo-se à determinação de minorantes para escolher, na amostra de S.A., a melhor solução encontrada. Para tal foi implementada a classe ***SimulatorRandomBag( )*** que recorre a uma lista para introduzir e remover os objetos *BBNode*. O ciclo termina quando se atinge o número de iterações, ou quando já não existirem nodos por ramificar.

Todas as pesquisas implementadas neste estudo, encontram-se ilustradas no pacote *Algorithms* com cor de fundo cinzenta no diagrama de classes do Anexo A.3.

2.4 Resultados computacionais

Nesta secção são apresentadas experiências computacionais com vista a distinguir, as vantagens e desvantagens, dos tipos de pesquisa implementados. Os resultados apresentados nesta secção contemplam o caso real da SAPEC Agro, em que uma operação pode ser processada em mais do que uma máquina e se admite recirculação. Nestas instâncias são então consideradas seis máquinas, as misturadoras, as *Jet Mill* 1 e 2 e a *Alpine*.

Os algoritmos foram implementados em Java e os testes foram realizados num computador com um processador Intel Core i5 e com 4 GB de memória RAM.

Na Tabela VI resumem-se os resultados computacionais feitos com instâncias fictícias de dimensões a variar entre 2 e 12 operações. Estas instâncias foram criadas, com o objetivo de ilustrar o desempenho das 3 pesquisas face a um aumento do número de tarefas. Como referido, o número de nós até ser encontrada uma solução é igual ao número de operações. Na formalização do modelo não foi admitida a recirculação e considerou-se que cada operação só pode ser realizada numa máquina. Repare-se que nesta formalização, a dimensão do espaço de S.A é dada por $(\sum_{i=1}^{3} O_{ij})!\ \ ;(j = 1, \dots n)$. Porém, apesar de estas instâncias serem fictícias, consideram 6 máquinas em vez das 3 sugeridas na formalização. Assim, foram consideradas na Tabela VI instâncias que variam entre 2 a 12 operações e, em cada instância, é considerado um lote por tarefa (dados da SAPEC- Agro).

Instância	*Depth First Search*			*Best First Search*			*Random Search*		
Nº * operações	t	*RAM*	S.O	t	*RAM*	*makespan*	t	*RAM*	*Makespan*
2	0.015	45k	220	0.001	36k	220	0.016	73k	220
3	0.001	56k	220	0.001	44k	220	0.001	74k	220
4	0.016	69k	220	0.016	45k	220	0.016	82k	220
5	0.109	55k	225	0.125	51k	225	0.109	79k	225
6	0.499	74k	275	0.561	59k	275	0.53	106k	275
7	0.812	81k	310	0.905	99k	310	0.811	146k	310
8	1.311	98k	315	3.291	247k	315	1.872	397k	315
9	4.867	124k	465	2.824	366k	465	13.868	443k	465
12	10.542	392k	490	x	x	x	18.325	637k	490

* Legenda: Consideram-se o total de operações para todos os produtos.

Tabela VI- Resultados computacionais (instâncias fictícias).

Como referido anteriormente, através do B&B-DFS foi possível obter a solução ótima (S.O.) em todas as instâncias da Tabela VI. A memória é constante. No entanto, para instâncias com 16 tarefas (testadas posteriormente a este estudo), o algoritmo ainda não tinha devolvido uma S.O. ao fim de 120 minutos. Por outro lado, repare-se que com o B&B-Best-FS, para instâncias com 12 operações o algoritmo utiliza a memória em disco para explorar o espaço de soluções admissíveis, tornando-se impraticável. Porém, com o B&B-RS foi possível encontrar uma S.O num tempo de resposta é aceitável para todas as instâncias. Nestas instâncias, este é o único algoritmo que se encontra limitado, a um máximo de 2. 000. 000 de iterações, por forma a não exceder as capacidades de memória deste computador. Para os restantes algoritmos a pesquisa é feita considerando toda a árvore. Destes resultados pode concluir-se que o B&B-Best-FS não é a melhor opção.

Os resultados para as instâncias reais, referentes aos dois meses de inverno deste ano na secção de WPs, estão resumidos na Tabela VII. Pretende-se comparar os valores dos tempos de produção (*makespan*) entre a solução implementada na fábrica e a solução que resultaria dos algoritmos B&B-DFS e B&B-RS. A solução implementada na fábrica é a que consta nos relatórios de produção realizada. Nestes relatórios existe informação respeitante ao número de máquinas em funcionamento e ao número de produtos produzidos. Deste modo, foi inserida na base de dados informação sobre os produtos encomendados, as máquinas disponíveis nessa semana, o horário laboral (sem feriados) e correram-se os algoritmos. Assim, apresentam-se instâncias reais que variam entre 45 e 127 operações. Os dois algoritmos encontram-se limitados ao mesmo número de iterações por forma a realçar as vantagens do B&B-RS face ao B&B-DFS.

SAPEC Agro			B&B -DFS			B&B Random Search		
Semana	Nº operações	*makespan*	*makespan*	Δ makespan (%)		t (seg)	*makespan*	Δ makespan (%)	
1	45	2520	1875	-	25,60	41.671	1790	-	28,97
2	127	4200	4095	-	2,50	35.678	3780	-	10,00
3	97	4200	2820	-	32,86	37.864	2745	-	34,64
4	118	4200	3730	-	11,19	39.429	3730	-	11,19
5	68	3360	2730	-	18,75	63.109	2670	-	20,54
6	104	4200	2490	-	40,71	37.955	2405	-	42,74
7	88	3360	2610	-	22,32	51.573	2570	-	23,51
8	105	4200	2890	-	31,19	46,426	2890	-	31,19
			$\bar{\Delta}$=	-	23,14		$\bar{\Delta}$=	-	25,35

Tabela VII- Resultados computacionais das primeiras 8 semanas do ano de 2013.

Durante as horas de almoço e jantar, em muitos dias destas semanas, as máquinas misturadoras ficavam a funcionar enquanto os trabalhadores almoçavam. No entanto, estes dados só se referem às horas de trabalho, ou seja, 14 horas por dia (dois turnos). Em algumas destas semanas existiram feriados e avarias nas máquinas que também foram tidos em conta. O *makespan* da SAPEC-Agro foi, assim, calculado de acordo com o número de minutos referente aos dias de trabalho de uma semana, considerando os feriados e o número de instalações disponíveis.

Para todas as semanas, o B&B-RS obteve soluções nunca piores que as encontradas com o B&B-DFS, facto que se deve à aleatoriedade na escolha dos nós a serem desenvolvidos. No entanto, não poderemos considerar nenhuma destas soluções como S.O uma vez que a árvore de B&B não é explorada na sua totalidade, por os algoritmos estarem limitados no número de iterações.

Para cada semana *s*, foi calculada a variação percentual dos tempos de produção, $\Delta_s = \frac{C_{max}\ algoritmo - C_{max}\ SAPEC}{C_{max}\ algoritmo} \times 100$.

A variação semanal média, $\bar{\Delta} = \frac{\sum_{s=1}^{8} \Delta_s}{8}$, do *makespan* nestas 8 semanas, da solução do B&B-RS face à praticada na fábrica, é de - 25,35%, ou seja, em média consegue-se uma redução de 17 horas e 44 minutos por semana, chegando a haver semanas em que a redução ultrapassa os 40%.

2.5 Conclusão

Neste estudo, aplicado à SAPEC Agro, é considerado um caso particular de um problema de *job shop* scheduling, uma vez que os tempos de *setup* dependem da ordem pela qual operações de diferentes famílias são processadas. Por forma a explorar o espaço de soluções admissíveis foi implementada a programação disjuntiva através de um algoritmo branch-and-bound (B&B), para o qual se desenvolveram três tipos de pesquisa em árvore (DFS, Best-FS, RS). Pretendia-se obter, se possível e num tempo útil, uma boa solução, pertencente à classe de escalonamentos imediatos, ou seja, uma solução que pudesse, com vantagem, ser utilizada pela SAPEC Agro.

Através da análise das experiências realizadas podemos afirmar que quando lidamos com muitas tarefas, como no caso da linha de produção de WPs, a melhor proposta de solução parece ser o B&B-RS. A procura de uma solução ótima com recurso ao B&B-DFS só se tornará praticável nos meses de verão, onde a procura é

reduzida. Para períodos de pico deve-se recorrer ao B&B-RS, por razões de memória e de desempenho.

Analisadas 8 semanas na seção de produção de WPs, no corrente ano de 2013, observa-se uma redução semanal média do tempo de produção (*makespan*) superior a 14 horas (2 turnos de trabalho), se se optar por implementar a solução proporcionada pelo algoritmo B&B-RS.

A SAPEC Agro poderá pois utilizar estes algoritmos para encontrar um bom, ou até, um ótimo escalonamento das encomendas. Para isto o utilizador conta com um Sistema de Apoio à Decisão que é detalhado no Anexo B. Poder-se-á assim melhorar a produtividade, reduzindo os custos e possibilitando um maior controlo dos prazos de entrega dos produtos, apoiando as atividades de planeamento e produção. Com o relatório de escalonamento obtido, os responsáveis da fábrica poderão ter também um maior controlo na alocação dos trabalhadores às máquinas. Pode, por exemplo, facilmente identificar e alocar trabalhadores com um bom desempenho a uma das máquinas cujo atraso afete o caminho crítico.

Bibliografia

ALLAHVERDI, A., NG, C. T., CHENG, T. C. E. & KOVALYOV, M. Y. 2008. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research,* 187**,** 985-1032.

ARTIGUES, C., BELMOKHTAR, S. & FEILLET, D. 2004. A New Exact Solution Algorithm for the Job Shop Problem with Sequence-Dependent Setup Times. *In:* RÉGIN, J.-C. & RUEHER, M. (eds.) *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems.* Springer Berlin Heidelberg.

ARTIGUES, C., BUSCAYLET, F. 2003. A fast tabu search method for the job-shop problem with sequence-dependent setup times. Proceedings of Metaheuristic International Conference, Kyoto, Japan. 1-6.

BALAS, E., SIMONETTI, N. & VAZACOPOULOS, A. 2008. Job shop scheduling with setup times, deadlines and precedence constraints. *Journal of Scheduling,* 11**,** 253-262.

BALLICU, M., GIUA, A. & SEATZU, C. 2002.. Job-shop scheduling models with set-up times. Systems, Man and Cybernetics, 2002 IEEE International Conference on, 6-9 Oct. 6 pp. vol.5.

CHEUNG, W. & ZHOU, H. 2001. Using Genetic Algorithms and Heuristics for Job Shop Scheduling with Sequence-Dependent Setup Times. *Annals of Operations Research,* 107**,** 65-81.

CHOI, I.-C. & CHOI, D.-S. 2002. A local search algorithm for jobshop scheduling problems with alternative operations and sequence-dependent setups. *Computers & Industrial Engineering,* 42**,** 43-58.

HASAN, S. M. K., SARKER, R. & ESSAM, D. 2010. Genetic algorithm for job-shop scheduling with machine unavailability and breakdowns. *International Journal of Production Research,* 49**,** 4999-5015.

J. PARK, D. L. 2012. Job Shop Scheduling with Job Families and Sequence-dependent Setups: Minimizing the Total Family Flow Time. *Proceedings of the Asia Pacific Industrial Engineering & Management Systems.* V. Kachitvichyanukul, H.T. Luong, and R.Pitakaso.

JOSEPH, B. J. H., THOMAS K. 1985. *EUROPEAN PATENT APPLICATION* [Online]. Available: https://data.epo.org/publication-server/rest/v1.0/publication-dates/19851113/patents/EP0160830NWA1/document.html.

NAUTA, C. J. 1979, http://www.google.com/patents/US4145144#forward-citations.

O'DONOVAN, R., UZSOY, R. & MCKAY, K. N. 1999. Predictable scheduling of a single machine with breakdowns and sensitive jobs. *International Journal of Production Research,* 37**,** 4217-4233.

PINEDO, M. L. 2012. *Scheduling: Theory, Algorithms, and Systems,* New York, Springer.

ROY, B. A. S., B. 1964. Les problèmes d'ordonnancement avec contraintes disjonctives. *Technical Report Note DS No. 9bis.* Paris: SEMA.

SCHRAGE, L. 1972. Solving Resource-Constrained Network Problems by Implicit Enumeration--Preemptive Case. *Operations Research,* 20**,** 668-677.

SPRECHER, A., KOLISCH, R. & DREXL, A. 1995. Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *European Journal of Operational Research,* 80**,** 94-102.

ZOGHBY, J., WESLEY BARNES, J. & HASENBEIN, J. J. 2005. Modeling the reentrant job shop scheduling problem with setups for metaheuristic searches. *European Journal of Operational Research,* 167**,** 336-348.

.

Anexos

Anexo A- Questões da implementação

Anexo A.1 A classe BBNode

Esta classe faz parte do pacote *BB_Colors* no diagrama de classes e apresenta os seguintes métodos:

- *Public* ***BBNode*** *(MasterTable mb),* construtor utilizado no primeiro nó (Master Node) - inicialização dos atributos do objeto;
- *Public* ***BBNode*** *( ),* construtor vazio utilizado nos restantes nós que inicialmente são uma cópia dos pais;
- *Public void* ***copyBBNode*** (BBNode old_branch, MasterTable mb) - responsável pela inicialização dos atributos como cópia dos atributos de um objeto nó "old_branch";
- *Public void* ***initializeMaster*** (MasterTable mb) - inicializa o primeiro nó (Master Node) com as informações das encomendas e preenche os sacos com o total das operações;
- *Public BBNode* ***genChild*** (MasterTable mb) - invocado no nó-pai para criação de nós descendentes. Retorna um nó que corresponde a um desenvolvimento possível a partir do nó pai e a alocação a efetuar corresponde à que está no topo da pilha de hipóteses do nó pai. O nó pai retira a hipótese efetuada (efetua *POP*) e quando a pilha finalmente se encontrar vazia, o nó pai já não gera mais filhos;
- *Public void* ***updateHypothesis*** (MasterTable mb) - chamado sempre que se cria um novo nó e gera todas as próximas alocações possíveis dado o estado do nó em questão. Estas hipóteses são armazenadas numa pilha com a finalidade de irem sendo retiradas à medida que cada uma é desenvolvida em filhos no método genChild();
- *Public boolean* ***isOver*** (MasterTable mb) - retorna verdadeiro se e só se o nó em causa não tiver mais operações para alocar;
- *Public void* ***delay*** ( ) - atrasa uma alocação e só é invocado quando não existem operações disponíveis nos sacos para ser executadas pela máquina que está na cabeça da fila de prioridade (PEC). Assim, acrescenta-se cinco minutos, ao

tempo de conclusão dessa máquina, de cada vez e até ser outra máquina a primeira a concluir a respetiva operação. O facto de se acrescentarem cinco minutos advém de todos os tempos de processamento nas máquinas da secção de WP's serem múltiplos de cinco;

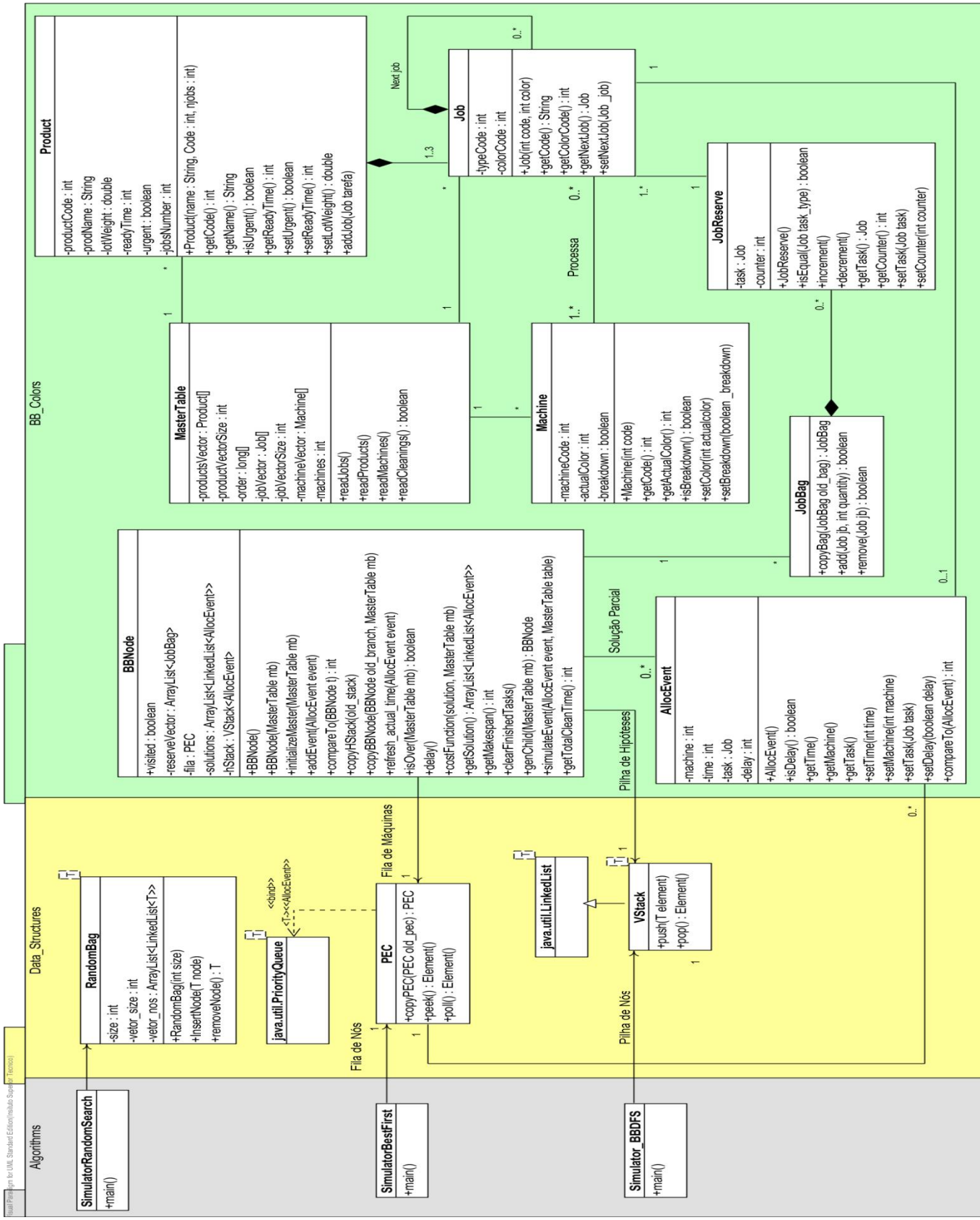
- *Public void* ***addEvent*** (AllocEvent event) - adiciona um evento já executado à solução atual, ou seja, à lista de alocações (uma lista para cada máquina);
- *Public static int* ***costFunction*** (solution, MasterTable mb) - retorna um inteiro que corresponde ao tempo de execução (em minutos) das operações alocadas desde o início até ao nó onde o método é invocado.
- *Private void* ***simulateEvent*** (AllocEvent event) - simula um evento no nó em que é invocado que consiste na afetação de uma operação a uma máquina. Uma operação é retirada do respetivo saco e a máquina é colocada na fila de espera (PEC), com o tempo de conclusão igual ao tempo atual somado com o de processamento da operação em causa e com o potencial tempo *setup*;

Anexo A.2 O algoritmo de Pinedo

Pseudocódigo do algoritmo 1 : Geração de todas as *afetações imediatas*

0) Inicialização
    - Assuma que Ω contém a primeira operação de cada tarefa;
    - Considere $T_{ij} = 0, \forall (i,j) \in \Omega$.
1) Seleção da Máquina
    - Determine para a afetação parcial $t(\Omega) = \min_{(i,j)\in\Omega} \{T_{ij} + p_{ij}\}$ e seja i^* a máquina na qual o mínimo é verificado.
2) Ramificação (*Branching)*
    - Assuma que Ω' representa o conjunto de todas as operações (i^*, j) a ser processadas na máquina i^* tal que $T_{i^*j} < t(\Omega)$;
    - Para cada tarefa em Ω' considere uma afetação parcial (estendida) com a próxima operação a ser alocada na máquina i^*. Para cada afetação parcial remover a operação de Ω, incluir a operação sucessora em Ω e voltar ao passo 2.

Anexo A.3 O diagrama de classes



Anexo B O Sistema

Por forma a englobar todas as restrições do escalonamento das encomendas da secção de formulação de WPs numa mesma ferramenta, otimizando, tanto quanto possível, o *makespan*, foi desenvolvida uma ferramenta que se divide em três sistemas: gestão da base de dados, gestão do modelo e interface gráfica.

A base de dados divide-se em dois sub-sistemas. Os **dados internos** representam a base de todas as metodologias implementadas. São estes que contêm os dados referentes às restrições da fábrica num ambiente de *JSP-SDST* ("*static_info"*). Os **dados do utilizador** abrangem todas as necessidades de produção e decisões que se enquadrem com as exigências do momento.

Este sistema é responsável pela manipulação entre a metodologia selecionada pelo utilizador, base de dados internos e decisões do utilizador, articulando-os por forma a resolver o problema de *JSP-SDST* com minimização do *makespan*.

A interface é a imagem do sistema. Esta é responsável por otimizar, tanto quanto possível, a interação entre o utilizador e a nova ferramenta. É nesta que se recebem os dados do utilizador e se apresenta o relatório de planeamento da produção.

Por ser um novo produto foi-lhe atribuído o nome de APASCH. Tal nome surgiu de uma junção da palavra Apache (tribo de índios americanos, conhecidos por serem guerreiros persistentes e aguerridos) com a palavra *scheduling*. Na realidade, trata-se de uma aplicação que envolveu um árduo trabalho de pesquisa e de aprendizagem e visa ser utilizada na afetação de tarefas. O logotipo da aplicação encontra-se na Ilustração 1.

Esta aplicação foi desenvolvida através da linguagem Java com o auxílio do programa *NetBeans IDE 7.3.1*, por forma a obter uma interface gráfica (*GUI*) eficiente e eficaz. É importante referir que no processamento da imagem se utilizou o *GIMP 2*. São ambos *freeware* (*software* gratuito).

Ilustração 1- Logotipo.

Na Ilustração 2, apresenta-se o janela do *login*. Esta é a primeira janela do programa e, uma vez itroduzidos os dados de *login* e validados, a aplicação abre a janela do "Menu" (como será ilustrado). É a partir da janela do "Menu", que o utilizador pode selecionar o botão *scheduling* sempre que pretenda realizar um planeamento da produção.

Ilustração 2- A janela do *login.*

Na Ilustração 3, apresenta-se a janela do menu. Esta imagem foi retirada pela minha câmara e o trabalhador fotografado autorizou a sua utilização neste sistema.

Ilustração 3- A janela do menu.

Na janela dos dados internos (ver Ilustração 4), o utilizador pode visualizar através de uma tabela todos os tempos de cada processo assumidos neste trabalho. A

origem destes tempos foi explicada no primeiro capítulo. A primeira coluna é referente ao produto e as restantes três representam os processos de Pré-Mistura, Moagem e Mistura-Final respetivamente Se o tempo de processamento de uma moagem depender da máquina a executá-lo, então, são visíveis tempos por máquina. Nesta janela insere-se ainda um botão com o nome "Menu", no campo superior direito, que permite ao utilizador, através de um clique, voltar ao Menu inicial.

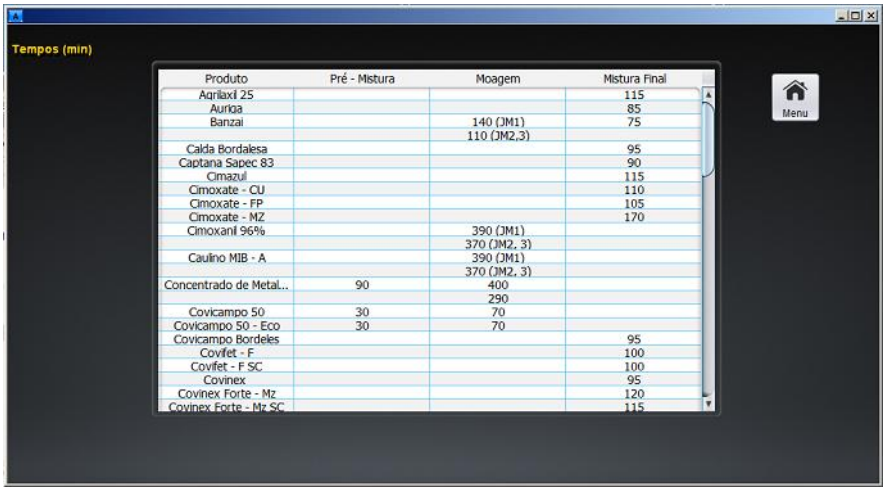


Ilustração 4- A janela dos dados internos.

Ao clicar no botão *"Scheduling"* a aplicação abre uma janela com o ambiente de trabalho do utilizador como mostra a Ilustração 5. É nesta janela que são introduzidos os dados do utilizador, que serão considerados pelo programa. Esta janela apresenta uma barra de menu, uma barra de ferramentas e um painel principal. Aqui, todos os componentes são bem visíveis, uma vez que apresentam tamanho, alinhamento e espaçamento igual.

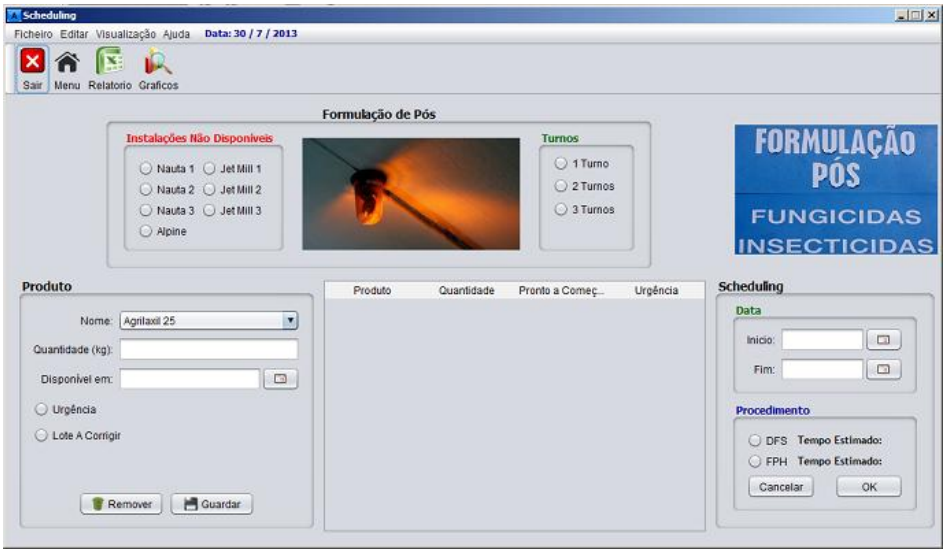


Ilustração 5- A janela *scheduling.*

Para facilitar a aprendizagem e compreensão na utilização da interface, foi criado um item de menu com o nome "ajuda". Aqui, é possível abrir um documento em *pdf* com todas as instruções necessárias e algumas ajudas para o funcionamento do APASCH.

A comunicação estabelecida pelo utilizador nesta janela (ver Ilustração 5) é apresentada na Ilustração 6.

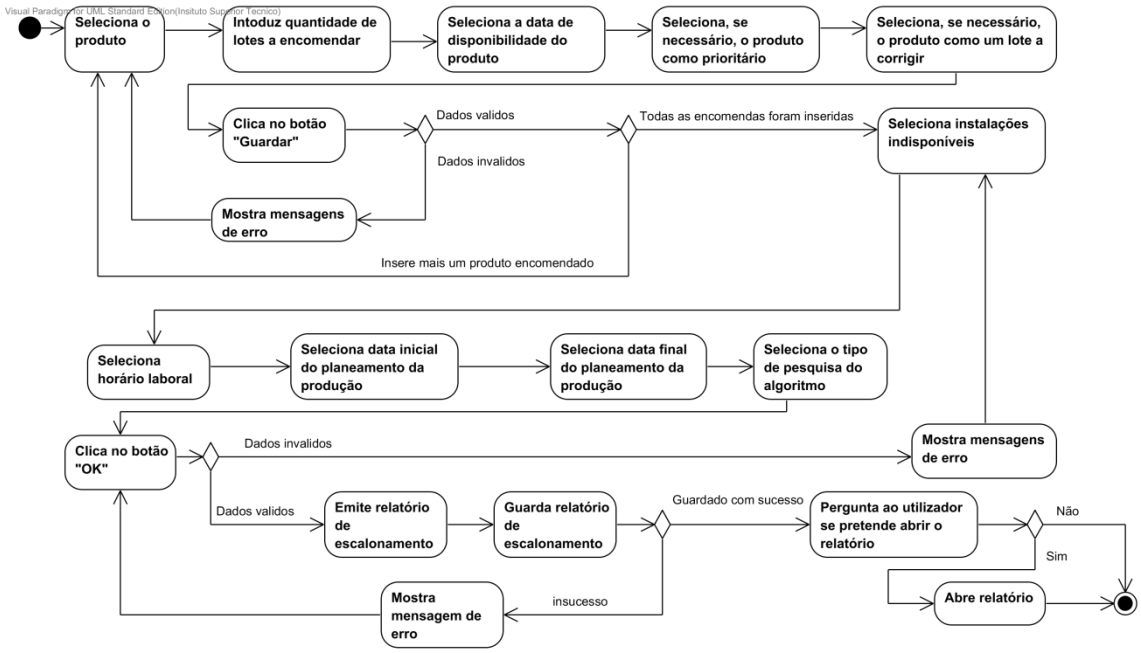


Ilustração 6- Fluxograma da interação do utilizador com a janela *"scheduling".*

Assim, a estrutura da janela *"scheduling"* (ver Ilustração 5) divide-se em 3 painéis, numa tabela e em duas imagens referentes a secção de formulação de WP's. Cada um dos sub-painéis refere-se a um domínio de restrições que o programa necessita de conhecer para produzir os resultados, sendo eles Formulação de Pós, Produto e *Scheduling* respetivamente.

No painel da Formulação de Pós (ver Ilustração 5), são visíveis dois sub-painéis. Assim, o utilizador deve selecionar através dos botões de opção quais as instalações da secção que, por motivo de avaria, ou de interesse não se encontrem disponíveis, assim como, o número de turnos em que esta secção está a laborar neste período em concreto.

No painel do Produto, visível na Ilustração 7, são selecionados os produtos a serem afetos às máquinas. Aqui, definem-se: a quantidade (em quilogramas), a data de disponibilidade (*ready time*) e, caso necessário, urgências e lote a corrigir. De forma a garantir que o utilizador não duvide que a sua ação foi registada pelo programa e, ainda, para possibilitar alteração ou eliminação de células, foi criada uma tabela que é atualizada sempre que um produto é selecionado.

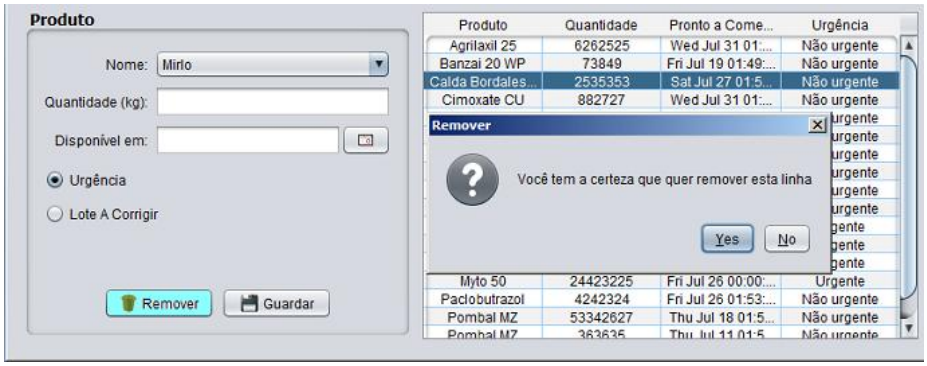


Ilustração 7- O sub-painel para as restrições que envolvem os produtos.

Uma vez que a introdução de dados pode, em muitos casos, ser exaustiva, foi implementada uma combo-box com os produtos da secção e instalado no NetBeans um ficheiro de um calendário em formato *jar*, como se mostra na Ilustração 8. Este ficheiro disponibiliza um calendário visível através da seleção do botão na data de disponibilidade. Por fim, dentro do painel de Scheduling, o utlizador introduz o intervalo temporal, para a qual o programa deve afetar as tarefas às máquinas, e escolhe a metodologia de otimização.

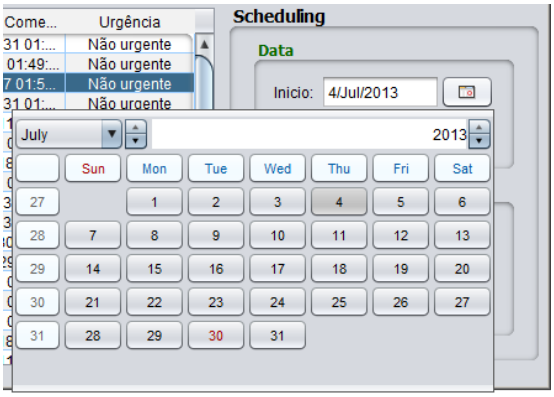


Ilustração 8- Calendário.

A aplicação encontra-se protegida da introdução de dados inesperados e condições perigosas. Através das caixas de diálogo de alerta (Ilustração 9), os utilizadores podem facilmente perceber como corrigir cada uma destas situações, através de mensagens precisas. Cada uma destas mensagens vem com um *icon* de alerta e fornece um comando de ajuda adicional, caso necessário.

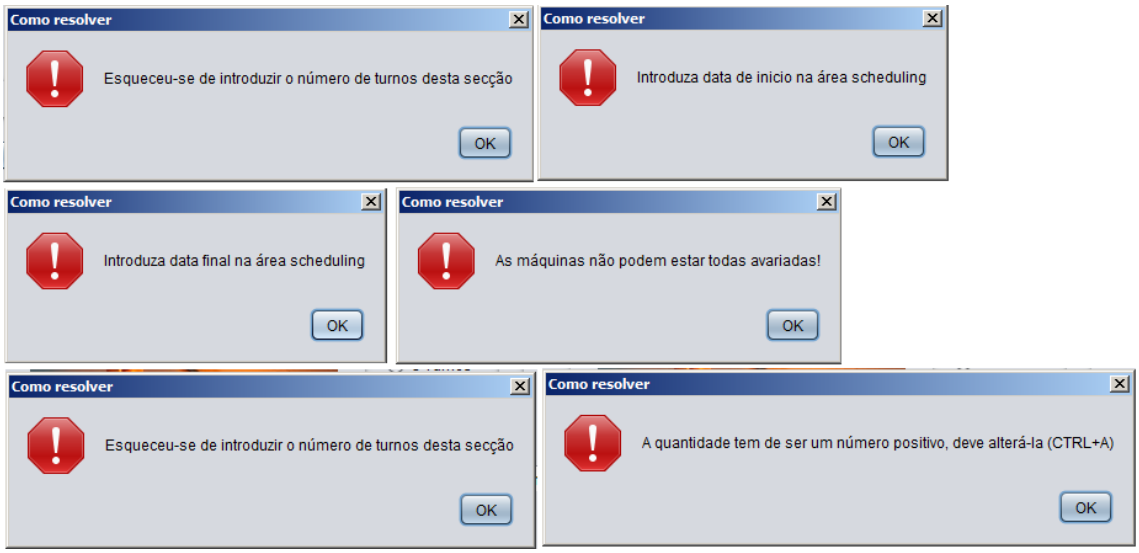


Ilustração 9- Mensagens de erro.

Para que o utilizador não duvide de que a sua ação teve uma reação, o *feedback* será transmitido quer através de caixas de diálogo informativas (observe a Ilustração 10), quer com a introdução da tabela visível na janela atrás mencionada.

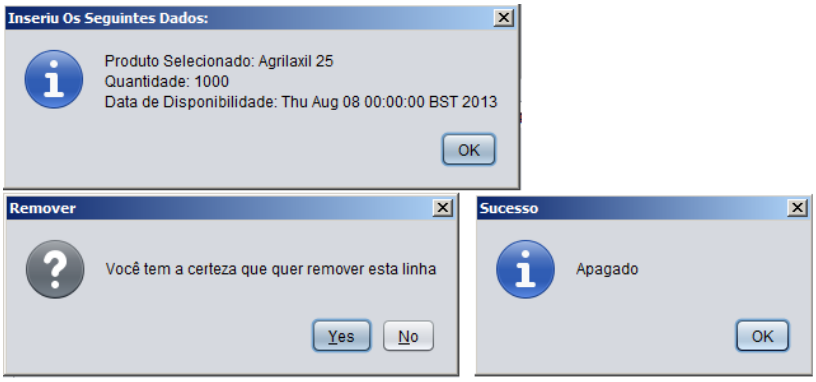


Ilustração 10- *Feedback* da tomada de decisões.