Instituto Politécnico de Viseu

Escola Superior de Tecnologia e Gestão de Viseu

Jorge Rafael Pereira Gomes

Otimização de corte 1-D

Tese de Mestrado

Sistemas e Tecnologias de Informação para as Organizações

Professor Doutor Francisco Ferreira Francisco

Professora Doutora Ana Cristina Matos

Setembro de 2012

“Pensar é o trabalho mais difícil que existe. Talvez por isso tão poucos se dediquem a ele.”

Henry Ford

Dedico este trabalho à minha futura esposa, Cátia Cardoso.

Como diz o seu avô: “sorte, saúde e gosto de viver”, por muitos e bons anos.

RESUMO

Face à atual conjuntura e às dificuldades que as empresas enfrentam diariamente para se manterem no ativo, cada vez mais a otimização e rentabilização de recursos é um fator de grande importância e decisivo.

Nesse sentido, o trabalho a realizar incide na otimização de cortes a 1 dimensão (1-D), aplicado concretamente à indústria de alumínios, no sentido de minimizar o desperdício de matéria-prima. Tipicamente o problema consiste em fazer a melhor gestão de corte das barras de material com dimensão padrão de 6,5 metros em pedaços mais pequenos para aplicação em portas, janelas, varandas, etc..

O objetivo é produzir uma ferramenta capaz de auxiliar na decisão do melhor plano de corte, sendo que o utilizador deve ter controlo sobre os resultados que pretende obter. Esta variação, distingue a solução a implementar de tantas outras que existem atualmente disponíveis, cuja função objetivo se centra unicamente em minimizar o desperdício de matéria-prima.

Parte do desafio passou por verificar a aplicabilidade de um método, nomeadamente o algoritmo genético, a um problema concreto e verificar os resultados produzidos. A implementação final recaiu na utilização da linguagem de programação PHP para a conceção do algoritmo, HTML e CSS para tratar da apresentação e introdução de dados, sendo acessível publicamente a partir do endereço http://jrgomes.net/corte1d.

Segundo a opinião do responsável da empresa que lançou o desafio e tem vindo a acompanhar e a utilizar a solução desenvolvida, os benefícios são claros, apresentando desde logo um ganho imediato em termos de rentabilidade, que aliado a uma melhor otimização do corte, face ao processo manual, pode culminar em lucros significativos ao longo do tempo.

A entrada de dados é bastante simples, e normalmente em menos de um segundo, a aplicação consegue apresentar visualmente as diversas boas soluções de corte, de acordo com o critério do utilizador.

De salientar que este produto se destina a empresas onde o serralheiro decide a seu belo prazer, face à experiência ou vícios obtidos ao longo do tempo, como efetuar os diversos cortes e não a industrias onde existem gabinetes de Engenheiros responsáveis por fazer chegar ao operador, as relações de corte devidamente otimizadas.

ABSTRACT

Due to the current situation and to the difficulties that companies are facing every day to keep in business, the optimization and maximization of resources is a matter of great importance and decisive.

Considering this, the work to be done is about the optimization of cuts in 1 dimension (1-D), specifically applied to the aluminum industry, with the purpose of reducing the waste of raw material. Typically the problem is about doing the best cut management of the material bars with a standard dimension of 6,5 meters into smaller parts to apply in doors, windows, balconies, etc.

The main purpose is to create a tool that will help deciding the best cut plan considering that the user must have control over the results to be achieved. This variation is what distinguishes the solution to be implemented, from so many others that are currently available but with the single purpose of minimizing the waste of raw material.

A part of the challenge was about verifying the applicability of a method, namely the genetic algorithm, to a specific problem, and verifying the achieved results. The final implementation used PHP programming language for the algorithm conception, HTML and CSS for the presentation and data introduction, being publicly accessed through http://jrgomes.net/corte1d.

According to the company's responsible, that proposed this challenge and that has been accompanying and using the developed solution, its benefits are obvious, and it presents an immediate gain in profitability that allied with a better optimization of the cut compared to the manual process can culminate in significant gains through time.

To emphasize that the data input is quite simple and that, in less than one second, the application can visually present several good cut solutions, according to the user criteria.

This product is destined to companies in which the metalworkers decide alone, considering their experience or the habits that were gained with time, how to perform several cuts, and not to industries where there are engineers responsible for giving the operator the cut relations duly optimized.

PALAVRAS CHAVE

otimização

corte

linear

1D

minimizar

desperdício

KEY WORDS

optimization

cut

linear

1D

minimize

waste

AGRADECIMENTOS

Uma tese de mestrado tem o reconhecido significado a nível académico, mas representa acima de tudo, o culminar de um enorme esforço e dedicação, sobretudo quando acumulado com os compromissos profissionais. O resultado deste trabalho, não teria sido conseguido sem o apoio de diversas pessoas que prestaram o seu preciso contributo.

Em primeiro lugar, quero agradecer ao Gilberto Figueiredo, proprietário de uma serralharia de alumínios na Figueira da Foz, que foi a pessoa responsável pelo desafio. Há cerca de 2 ou 3 anos que me abordava com a questão de realizar uma aplicação que satisfizesse as suas necessidades, no que diz respeito à otimização e planeamento de cortes. A tese de mestrado foi o momento certo para agarrar o desafio e apresentar a solução. Mais de 30 anos de experiência no ramo foram decisivos para ir dando diversas dicas práticas e validar os resultados.

Quero também deixar uma mensagem de agradecimento aos meus orientadores, Prof.º Dr.º Francisco Ferreira Francisco e Prof.ª Dr.ª Ana Cristina Matos, pela disponibilidade, sugestões e dinâmica incutida na realização desta dissertação.

Agradecer também a todos os familiares próximos e amigos que, de forma direta ou indireta, deram o seu apoio na realização desta tese.

Finalmente agradecer à minha noiva, Cátia Cardoso, por todo o apoio e paciência que teve em me aturar nesta fase, véspera de casamento.

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE QUADROS

1. Introdução

Problemas de corte unidimensionais (1D) estão presentes em diversos setores da indústria onde existe a necessidade de se cortar peças maiores em peças menores. Existem alguns algoritmos exatos para encontrar a solução ótima, a título de exemplo ver [Vanderbeck, F. 2000; Valério de Carvalho, 1999], mas devido a limitações computacionais de tempo e/ou espaço, podem ser aplicados apenas a problemas de pequena dimensão e, por isso, não serão aqui analisados, já que existe apenas interesse em encontrar soluções boas e rápidas, sabendo à partida que a ótima não é garantida.

Há milhões de maneiras diferentes para se cortar as peças de um perfil com dimensões padronizadas. Num optimizador de corte, informam-se as dimensões dos perfis disponíveis e as dimensões das peças a serem cortadas. O algoritmo calcula automaticamente a posição de corte para cada peça, de maneira que a sobra de material seja a menor possível.

Para testar todas as possibilidades de corte para N peças seria necessário efetuar o cálculo N! (N Fatorial). Operação bastante simples, rápida e eficaz, para calcular o corte com 3 a 6 peças, considerando que 6! representa 720 combinações possíveis. No entanto, ao passar para uma escala ligeiramente superior em que temos por exemplo 20 peças diferentes, seria necessário calcular 20!, isto é, qualquer coisa como 2.432.902.008.176.640.000 soluções possíveis de corte. Tal situação é impossível de calcular manualmente e demasiado morosa para calcular computacionalmente, não sendo possível obter uma solução ótima imediata.

Uma empresa real que opera basicamente na construção de portas, janelas, marquises e varandas em alumínio, mas que serve de exemplo a tantas outras com necessidades similares, necessita de uma solução de baixo custo para otimizar o corte de peças.

Eis um exemplo de um caso concreto:

A maioria dos perfis de alumínio utilizados têm 6,50m de comprimento, mas podem ser diferentes, logo esta opção deve ser configurável. Da relação de uma obra resultam as seguintes necessidades:

4 x 1,80m + 6 x 0,90m + 3 x 2,50m + 8 x 1,30m + 5 x 1,00m + …

Quantos perfis de 6,50m serão precisos, fazendo a melhor gestão de corte?

Uma possível solução feita a olho e atendendo à experiência do funcionário pode ser:

- 1 Perfil de 6,50 -> (2 x 2,50) + (1 x 1,30) sobra 0,20 de desperdício
- 1 Perfil de 6,50 -> (3 x 1,80) + (1 x 1,00) sobra 0,10 de desperdício
- 1 Perfil de 6,50 -> (6 x 0,90) + (1 x 1,00) sobra 0,10 de desperdício
- …

Com poucas peças e medidas pequenas a solução torna-se relativamente fácil de executar “a olho”. Com muitas peças, chegar a uma boa solução, com este método, complica-se. O que habitualmente o serralheiro da empresa em causa faz, é começar sempre por cortar as peças mais compridas, tendo sempre a preocupação com a ponta de desperdício: se for curta a sobra, vai para o lixo, se tiver mesmo de sobrar que tenha um comprimento que tenha utilidade para a obra seguinte, sempre superior a 80cm, e assim já não se considera desperdício, mas sim uma sobra para a obra seguinte.

Algumas considerações:

- Um desperdício máximo de 0,30m é aceitável (lixo).
- Um desperdício entre 0,30m e 0,80m já é crítico, também é quase lixo mas caro, pois pouco aproveitamento dá para obras futuras.
- Um desperdício superior a 0,80m já continua a ser aceitável, pois guarda-se e serve para a obra seguinte.

Acresce ainda uma outra variável ao problema que pode alterar completamente o propósito do algoritmo, dependendo da especificidade do material a cortar. Isto é, se o material a cortar for amplamente utilizado, efetivamente pode-se dar o caso em que é desejável ter sobras de maior dimensão para futuras obras em detrimento de sobras de pequena dimensão que são consideradas lixo. Mas se o material a cortar for específico para uma dada obra, por exemplo alumínio lacado a vermelho, existe todo o interesse em fazer o melhor aproveitamento possível, minimizando o desperdício total dos cortes, pois dificilmente tem utilidade para outras obras.

Antes de se avançar para as abordagens a considerar na resolução deste problema, é importante ter em conta alguns conceitos fundamentais que vão permitir entender e contextualizar a natureza do mesmo, apresentados no capítulo seguinte.

1.1 Objetivos

Da opinião recolhida junto de alguns serralheiros locais, proprietários ou funcionários de empresas de dimensão reduzida, a complexidade das soluções existentes no mercado é o principal fator, seguido do preço, para que tais não sejam utilizadas. Desta forma, o principal desafio não é obter soluções iguais ou melhores mas sim proporcionar uma interface simples mas funcional que permita a qualquer leigo em tecnologias uma utilização bastante intuitiva e natural.

Os principais objetivos a atingir são:

1. Interface simples, minimalista e amigável, muito próxima à relação tirada na obra, tendo em vista o público-alvo (serralheiros);
2. Solução online, que permite tirar partido do poder de processamento dos servidores web e acessível a partir de qualquer dispositivo com acesso à internet;
3. Utilização gratuita;
4. Controlo sobre o algoritmo no sentido de minimizar o desperdício ou maximizá-lo, para que as sobras sejam úteis para obras futuras;
5. Possibilidade de apresentar graficamente o plano de corte, otimizado para impressão;
6. Idioma em Português mas com a possibilidade de internacionalização;

7. Tirar partido de recursos atuais no paradigma da web, tais como HTML5 e CSS3.

1.2 Estado da Arte

No que diz respeito ao estado da arte, obviamente existem já muitas soluções comerciais para o problema. A solução final a apresentar pretende ser diferente nos seguintes pontos: Interface simples, minimalista e amigável, muito próxima à relação tirada na obra, tendo em vista o público-alvo (serralheiros); Solução online, que permite tirar partido do poder de processamento dos servidores web e acessível a partir de qualquer dispositivo com acesso à internet; Uso gratuito; Controlo sobre o algoritmo no sentido de minimizar o desperdício ou maximizá-lo, para que as sobras sejam úteis para obras futuras; Possibilidade de apresentar graficamente o plano de corte, otimizado para impressão.

No que diz respeito a aplicações web, e à data de escrita deste documento apenas foi encontrada uma solução (1d-cut-optimization), mas que apresenta ainda alguma complexidade na sua utilização, como se pode observar pela descrição dos passos necessários, disponíveis no anexo 1.

De outras soluções existentes para Windows, é possível destacar aquelas que foram alvo já de testes, para se compreender o seu funcionamento e que resultados permitem obter, que são:

- Corte Certo 1D, disponível em http://www.cortecerto.com/
- 1D-Nesting Optimizer, disponível em http://www.1d-solutions.com/
- Bar Cut Optimizer, disponível em http://www.binrace.com/bar-cut-optimizer/
- Cut 1D X, disponível em http://www.optimalprograms.com/cut_1d_x.htm
- CutLogic 1D, disponível em http://www.tmachines.com/index.htm
- PLUS 1D, disponível em http://www.nirvanatec.com/bar_nesting_software.html

É de realçar que a 1ª (Corte Certo 1D) está disponível em vários idiomas sendo o Português um deles.

O propósito das aplicações deste segmento é efetivamente minimizar o desperdício apresentando sempre a melhor solução nesse sentido. No entanto e de acordo com os cerca de 30 anos de experiência do serralheiro que lançou este desafio, e como também já foi referido, nem sempre um plano de corte com menor desperdício é a melhor solução.

A melhor solução depende também da especificidade do material em uso. Se o material em questão for amplamente utilizado, pode ser mais benéfico deixar segmentos com comprimentos maiores (sempre superiores a 80cm) do que ter sobras de 30 ou 40cm, uma vez que as maiores têm sempre utilização em obras futuras. Deste modo um desperdício de uma peça de 80cm é preferível a duas peças de 30cm que no total somam 60cm de desperdício.

Por outro lado, se o material a cortar for demasiado específico para uma dada obra, não existe qualquer interesse em guardar pontas para aproveitamentos futuros, pois a probabilidade de reutilização deste tipo de material é muito escassa. Este cenário enquadra-se na abordagem clássica do problema de corte 1-D, pois o objetivo será minimizar o desperdício total.

Como definido nos objetivos, o que se pretende é ser possível ter algum controlo sobre o algoritmo no sentido de influenciar o tamanho das pontas de sobra e deixar que o utilizador, de acordo com toda a sua experiência, decida qual é efetivamente a solução mais adequada. Das soluções analisadas até ao momento, nenhuma delas fornece tal funcionalidade, neste sentido, a aplicação a desenvolver distingue-se da abordagem clássica do problema de corte 1-D, tratando-se de uma extensão da mesma.

1.3 Método

Face ao exposto é possível constatar que estamos perante um cenário de criação e conceção de uma solução, para uma extensão do problema de corte 1-D, cuja principal fonte foi a sugestão de um serralheiro que opera no mercado há cerca de 30 anos.

Claro que o primeiro passo foi tentar compreender a real necessidade e perceber se não existem efetivamente soluções no mercado que respondam a esse propósito. Com isto foram já analisadas algumas das soluções mais populares apresentadas pelos motores de busca e que constam no tópico anterior (estado da arte).

Posteriormente era importante perceber o que se passa em empresas de maior dimensão. Foi contactada uma das maiores empresas da região (Martifer) que facultou uns exemplos (ver anexos) de como abordam o problema à sua escala que claramente está totalmente desajustada das meras necessidades sentidas por serralheiros nas suas micro ou pequenas empresas.

O passo seguinte foi contatar a AIRV (Associação Empresarial da Região de Viseu), no sentido de poder aceder ao seu diretório de empresas. Serão contactadas empresas do setor, no

sentido de reunir mais informação sobre as necessidades ou soluções que cada uma apresenta para este problema em concreto.

1.4 Estrutura da Dissertação

Este documento encontra-se estruturado da seguinte forma:

- No segundo capítulo são explanados de forma sucinta conceitos chave relacionados com a ordem de complexidade de problemas de otimização combinatória e os algoritmos genéticos, que permitem uma melhor compreensão, quer do problema, quer da solução implementada.
- Os métodos utilizados na abordagem ao problema são descritos no capítulo três.
- O capítulo quarto tem como objetivo apresentar as soluções encontradas e sua implementação. É explicado com detalhe como a solução foi evoluindo e as etapas percorridas.
- No capítulo cinco são demonstrados os resultados obtidos e efetuadas comparações com outras soluções comerciais.
- Por fim, no sexto capítulo, é efetuada uma súmula das principais características da solução onde se apresentam os resultados face aos objetivos traçados. São tiradas conclusões bem como a indicação de possíveis melhorias a desenvolver no futuro.

2. Contextualização

É necessário compreender alguns conceitos chave antes de se avançar com mais detalhes sobre o problema, tais como:

Um algoritmo é um método finito para resolver um dado problema.

Analisar um algoritmo é determinar a quantidade de recursos (como tempo e armazenamento) que são necessários para executá-lo [Wikipédia – Análise de algoritmos].

A complexidade de um algoritmo consiste na quantidade de "trabalho" necessária para a sua execução, expressa em função das operações fundamentais, as quais variam de acordo com o algoritmo, e em função do volume de dados.

- Um algoritmo, como já vimos, serve para resolver um determinado problema, e todos os problemas têm sempre uma entrada de dados.
- O tamanho dessa entrada (n) tem geralmente efeito direto no tempo de resposta do algoritmo.
- Dependendo do problema a ser resolvido, é provável existirem já algoritmos prontos ou que podem ser adaptados.
- O problema é: qual o algoritmo a escolher?

Como comparar soluções para o mesmo problema?

- Vamos supor que temos dois métodos para localizar um elemento num vetor: Linear (LIN) e Binário (BIN)
- Temos ainda dois computadores diferentes: A – Core 2 Duo e B – Celeron

Pela análise do quadro 1, a tendência imediata, será indicar que a pesquisa num vetor linear no computador A é a melhor solução visto ser muito mais rápida.

Quadro 1 – Comparação de tempos de execução – poucos elementos

Tamanho (n)	Tempo de execução de LIN em A	Tempo de execução de BIN em B
16	8 ns	100.000 ns
64	32 ns	150.000 ns
256	128 ns	200.000 ns
1024	512 ns	250.000 ns

No entanto, uma abordagem superficial, com poucos dados (n), nem sempre demonstra o comportamento da solução perante um volume de dados significativamente maior.

O quadro 2 mostra a evolução dos resultados, para diferentes tamanhos de (n).

Quadro 2 – Comparação de tempos de execução – muitos elementos

Tamanho (n)	Tempo de execução de LIN em A	Tempo de execução de BIN em B
16	8 ns	100.000 ns
64	32 ns	150.000 ns
256	128 ns	200.000 ns
1024	512 ns	250.000 ns
...	...	...
1.048.576	524.288 ns	500.000 ns
4.194.304	2.097.152 ns	550.000 ns
16.777.216	8.388.608 ns	600.000 ns
...	...	...
$63{,}072 \times 10^{12}$	$31{,}536 \times 10^{12}$ ns ou **1 ano**	1.375.000 ns ou cerca de **1,4 segundos**

Uma análise cuidada ao quadro 2 revela que a conclusão obtida anteriormente deixa de fazer sentido. Isto é, já não é de todo verdade que uma pesquisa linear executada no computador A

é a solução que permite obter o resultado mais rapidamente, pois depende do tamanho de dados que estejam a ser tratados.

Este é o verdadeiro dilema, pois por vezes julga-se ter uma boa solução, quando o conjunto de dados de entrada é diminuto e normalmente nem existe a tendência para examinar o comportamento para volumes de dados maiores.

É de evitar soluções que tenham um comportamento exponencial. A figura 1 mostra a tendência de cada uma das funções em função do tamanho (n), sendo evidente que a função representada a azul tem um desempenho pior.

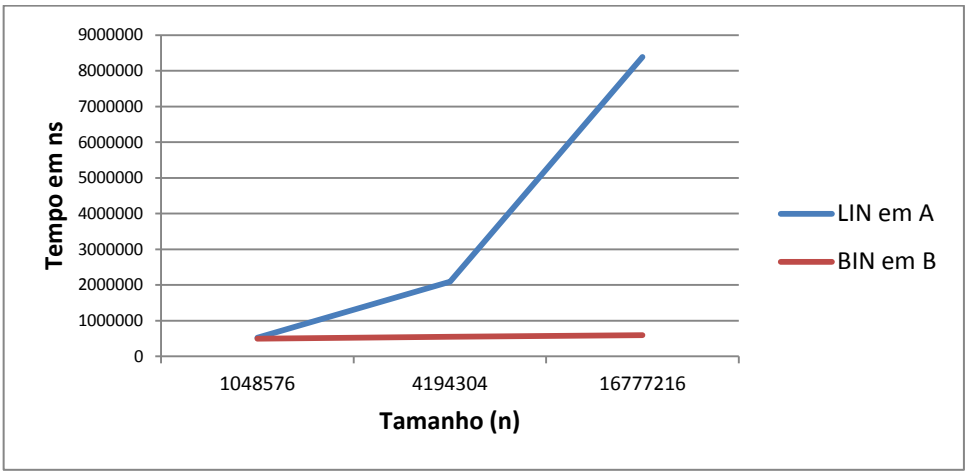


Figura 1 – Análise gráfica entre soluções LIN em A, e BIN em B

Perante os factos apresentados e para volumes significativos de dados, a diferença que existe entre a capacidade de processamentos de ambos os computadores não se sobrepõe à eficiência dos algoritmos, sendo que uma pesquisa binária permite obter mais rapidamente um resultado, comparativamente à pesquisa linear.

Outro exemplo simples para melhor elucidar esta questão é o seguinte:

Considere o número de operações de cada um dos dois algoritmos, que resolvem o mesmo problema, como função de n.

- Algoritmo 1: $f1(n) = 2n^2 + 5n$ operações
- Algoritmo 2: $f2(n) = 500n + 4000$ operações

Dependendo do valor de n, o Algoritmo 1 pode necessitar mais ou menos operações que o Algoritmo 2, tal como demonstra o quadro 3.

Quadro 3 – Comparação das funções para diferentes valores de n

n	f1	f2
10	250	9000
100	20500	54000
500	502500	254000
1000	2005000	504000

Um caso de particular interesse é quando n tem valor muito grande ($n \rightarrow \infty$), denominado comportamento assintótico. Este comportamento determina que termos de menor grau podem ser desprezados:

- $n^2 + n$ será aproximado de n^2
- $6n^3 + 4n – 9$ será aproximado para n^3

Os termos inferiores e as constantes multiplicativas contribuem pouco na comparação e podem ser descartados. O importante é observar que f1(n) cresce com n^2 ao passo que f2(n) cresce com n. Um crescimento quadrático é considerado pior que um crescimento linear. Assim, vamos preferir o Algoritmo 2 ao Algoritmo 1.

No que toca à complexidade dos algoritmos, podemos falar de dois tipos:

- Complexidade espacial – Quantidade de recursos utilizados para resolver o problema;
- Complexidade temporal – Quantidade de tempo utilizado. Pode ser vista como o número de instruções necessárias para resolver um determinado problema ou mesmo o tempo real de execução;

Nas duas situações, a complexidade é medida de acordo com o tamanho dos dados de entrada (n). Para o estudo da complexidade são usados três perspetivas: Pior cenário; Melhor cenário; Cenário médio.

O Pior cenário é normalmente representado por O( ). Por exemplo se dissermos que um determinado algoritmo é representado por g(x) e a sua complexidade para o pior cenário é n, será representado por g(x) = O(n). Basicamente consiste em assumir o pior dos casos que pode acontecer, sendo muito usado e normalmente o mais fácil de determinar.

Exemplo: Se existirem quatro caixas, sendo que apenas uma delas tem algo dentro e as restantes estão vazios a complexidade para o pior cenário será O(4) pois no pior dos casos encontra-se a caixa com conteúdo à quarta tentativa.

O Melhor cenário representa-se por Ω ( ). Método que consiste em assumir que vai acontecer o melhor. É pouco usado e tem aplicação em poucos casos.

Exemplo: Se tivermos uma lista de números e quisermos encontrar algum deles assume-se que a complexidade para o melhor cenário é Ω (1) pois assume-se que o número estaria logo na cabeça da lista.

O Cenário médio representa-se por θ( ). Este método é dos três o mais difícil de determinar pois necessita de análise estatística e como tal muitos testes. No entanto é muito usado pois é também o que representa mais corretamente a complexidade do algoritmo.

Outro conceito a saber são os limites (superior e inferior) de um algoritmo, também designados de *upper bound* e *lower bound*.

O Limite superior permite identificar a complexidade máxima para determinado problema:

- Seja dado um problema, como multiplicação de duas matrizes quadradas n x n;
- É conhecido um algoritmo para resolver este problema (pelo método trivial) de complexidade $O(n^3)$;
- Sabemos assim que a complexidade deste problema não deve superar $O(n^3)$, uma vez que existe um algoritmo que o resolve com esta complexidade;
- O limite superior deste problema é $O(n^3)$;
- A cota superior de um problema pode mudar se alguém descobrir um outro algoritmo melhor;
- O Algoritmo de Strassen reduziu a complexidade para $O(n^{\log 7})$. Assim o limite superior do problema de multiplicação de matrizes passou a ser $O(n^{\log 7})$;
- Coppersmith e Winograd melhoraram ainda para $O(n^{2.376})$;

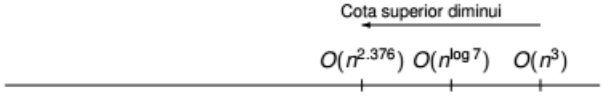


Figura 2 – Limite superior

- O limite superior para resolver um problema pode ser comparado com o record mundial de uma modalidade de atletismo. É estabelecido pelo melhor atleta (algoritmo) do momento.

- Assim como um record mundial, o limite superior pode ser melhorado por um algoritmo (atleta) mais veloz.
- Ex: corrida 100 metros: 1988 Carl Lewis (9.92seg); 2009 Usain Bolt (9.58seg)

O limite inferior, por vezes permite demonstrar que para um dado problema, qualquer que seja o algoritmo a ser usado, o problema requer pelo menos um certo número de operações:

- Para o problema de multiplicação de matrizes quadradas n x n, apenas para ler os elementos das duas matrizes de entrada ou para produzir os elementos da matriz produto leva tempo $O(n^2)$. Assim uma cota inferior trivial é (n^2);
- Na analogia anterior, uma cota inferior de uma modalidade de atletismo não dependeria mais do atleta. Seria algum tempo mínimo que a modalidade exige, qualquer que seja o atleta. Uma cota inferior trivial para os 100 metros rasos seria o tempo que a velocidade da luz leva para percorrer 100 metros no vácuo;
- Se um algoritmo tem uma complexidade que é igual ao limite inferior do problema, então ele é assintoticamente ótimo;
- O algoritmo de Coppersmith e Winograd é de $O(n^{2:376})$ mas a cota inferior (conhecida até hoje) é de (n^2). Portanto não podemos dizer que ele é ótimo.

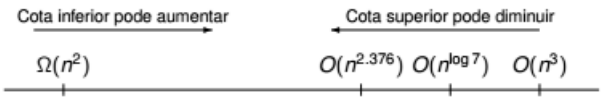


Figura 3 – Limite inferior e superior

2.1 Ordens de complexidade

Dependendo da complexidade do algoritmo, podemos assim classificá-lo. Para se perceber melhor o que a ordem de complexidade representa, vamos ver alguns exemplos. De notar, que para se calcular a complexidade de um algoritmo, deve-se analisar o pior caso possível.

Imaginando que um computador leva 1ms a processar uma operação, o quadro 4 indica o tempo aproximado de execução de um algoritmo com diferentes ordens de complexidade para diferentes tamanhos de entrada.

Quadro 4 – Ordens de complexidade

n	O(n)	Log(n)	nLog(n)	$O(n^2)$	$O(n^3)$	$O(2^n)$	O(n!)
16	0.016s	0.004s	0.064s	0.256s	4s	1m5s	663 anos
32	0.032s	0.005s	0.16s	1s	33s	49 dias	10^{23} sec
512	0.512s	0.009s	4.608s	4m22s	37h	10^{143} sec	...

Função de custo	Tamanho n					
	10	20	30	40	50	60
n	0,00001 s	0,00002 s	0,00003 s	0,00004 s	0,00005 s	0,00006 s
n^2	0,0001 s	0,0004 s	0,0009 s	0,0016 s	0,0.35 s	0,0036 s
n^3	0,001 s	0,008 s	0,027 s	0,64 s	0,125 s	0.316 s
n^5	0,1 s	3,2 s	24,3 s	1,7 min	5,2 min	13 min
2^n	0,001 s	1 s	17,9 min	12,7 dias	35,7 anos	366 séc.
3^n	0,059 s	58 min	6,5 anos	3855 séc.	10^8 séc.	10^{13} séc.

Fonte: ZIVIANI, Nívio. Projetos de Algoritmos

Sempre que a aplicação desenvolvida apresentar sintomas de demora na apresentação de resultados é necessário ponderar se o problema reside nas limitações físicas, havendo necessidade de substituir hardware, ou se o algoritmo deve ser revisto numa tentativa de efetuar otimizações.

Todos os elementos aqui apresentados, visam transmitir o seguinte pensamento antes de começar a desenvolver a lógica da aplicação: Ponderar sempre antes de colocar um ciclo dentro de outro, sejam eles do tipo “for” ou “while” e verificar se vai percorrer todo um ciclo até ao final sem necessidade. Coisas tão simples quanto estas, podem ter um impacto muito significativo no desempenho do algoritmo.

2.1.1 Complexidade - O(1)

São os algoritmos onde a complexidade é independente do tamanho n de entradas. É o único em que as instruções do algoritmo são executadas um número fixo de vezes.

```
if (condição == true) then {
        realiza alguma operação em tempo constante
}
else  {
        realiza alguma operação em tempo constante
}
```

Figura 4- Complexidade O(1)

2.1.2 Complexidade linear - O(N)

Uma operação é realizada em cada elemento de entrada. Ex: pesquisar elementos numa lista.

```
for (i = 0; i < N; i = i + 1 ) {
        if (condição == true) then {
                realiza alguma operação em tempo constante
        }
        else  {
                realiza alguma operação em tempo constante
        }
}
```

Figura 5 – Complexidade linear

2.1.3 Complexidade Logarítmica - O(LogN)

Ocorre tipicamente em algoritmos que dividem o problema em problemas menores.

```
int PesquisaBinaria ( int array[], int chave , int N){
        int inf = 0;
        int sup = N - 1;
        int meio;
        while (inf <= sup) {
                meio = (inf+sup)/2;
                if (chave == array[meio]) return meio;
                else
                        if (chave < array[meio]) sup = meio-1;

                        else inf = meio+1;
        }
        return -1;   // não encontrado
}
```

Figura 6 – Complexidade logarítmica

2.1.4 Complexidade Quadrática - O(N^2)

Instruções são processadas aos pares, geralmente com um ciclo dentro de outro.

```
void bubbleSort(int[] a) {
        for (int i = 0; i < a.length-1; i++) {
                for (int j = 0; j < a.length-1; j++) {
                        if (a[j] > a[j+1]) {
                                swap(a, j, j+1);
                        }
                }
        }
}
```

Figura 7 – complexidade quadrática

2.1.5 Complexidade Cúbica - O(N^3)

Instruções são processadas três a três, vulgarmente com um ciclo dentro de outros dois.

```
int dist[N][N];
int i, j, k;

for ( k = 0; k < N; k++ )
        for ( i = 0; i < N; i++ )
                for ( j = 0; j < N; j++ )
                        dist[i][j] = min( dist[i][j], dist[i][k] +
                                                      dist[k]
[j] );
```

Figura 8 – Complexidade cúbica

2.1.6 Complexidade Exponencial – O(2^n)

Utilização do método habitualmente denominado de “força bruta”, que visa explorar todas as possíveis combinações para encontrar a solução do problema. A solução geralmente é baseada diretamente no enunciado do problema e nas definições dos conceitos envolvidos.

Exemplo: Utilizando apenas números é possível criar 10^n senhas de n dígitos. Um algoritmo de força bruta para quebrar uma dessas senhas tem complexidade O(2^n).

2.1.7 Complexidade Fatorial – O(N!)

Também é baseado na utilização de procura exaustiva para encontrar a solução de um problema. Consiste em testar todas as combinações possíveis existentes à procura da solução ótima para o problema.

Um exemplo típico é o problema do "Caixeiro viajante" que tem por objetivo encontrar a rota mais curta para visitar várias cidades, sem repetir nenhuma delas. Este tipo de problema, à semelhança do problema aqui em estudo, não possui uma solução exata e eficiente, sendo assim considerado um problema NP-hard[1].

Se o problema for resolvido pela abordagem "força bruta" (analisar todas as combinações de peças possíveis e depois escolher o melhor), a complexidade deste problema pode ser dada por N! (N fatorial), onde N representa o número de peças, pelo que este problema é considerado como NP-hard [Garey e Johnson 1979; Dyckhoff, 1990].

Por causa desta e de outras características, este é um dos problemas de otimização combinatória mais estudados na literatura, tendo diversas abordagens de solução propostas [Johnson e McGeoch 1997].

2.2 Algoritmo genético

Antes de mais interessa explicar alguns conceitos chave que são de vital importância para a compreensão deste trabalho, tais como informações sobre os Algoritmos genéticos (AG), a sua estrutura, o seu modo de funcionamento e os seus constituintes, uma vez que vai ser o método utilizado para solucionar o problema em questão.

2.2.1 O que é um algoritmo genético?

Os AG's são técnicas de procura e otimização baseados em mecanismos de seleção natural. Foram desenvolvidos por John Holland em conjunto com os seus colegas e alunos na Universidade de Michigan nas décadas de 60 e 70, com o objetivo de estudar formalmente o processo de adaptação natural, implementando-os em sistemas computacionais [Holland 1975]. Estes algoritmos permitem abordar problemas complexos sobre os quais exista pouca

[1] Consultar http://en.wikipedia.org/wiki/NP-hard para mais detalhes.

informação, exceto a forma de avaliar uma boa solução. Podem ser aplicados a um conjunto diverso de problemas [Goldberg 1989], permitindo obter melhores resultados que outras técnicas de otimização tradicionais.

Um AG é um algoritmo não determinista, que procura soluções aproximadas, usando técnicas derivadas da biologia (tais como herança, cruzamentos, recombinação, mutação e o princípio da seleção natural proposto por Darwin) para problemas que não são exequíveis em tempo útil usando técnicas de resolução, tais como métodos matemáticos, método simplex ou outros métodos que visam encontrar a melhor solução assentando em funções determinísticas.

Um método determinístico devolve sempre o mesmo resultado perante um conjunto específico de valores de entrada, enquanto que um método não determinístico pode devolver resultados diferentes a cada vez que é invocado, para o mesmo conjunto de valores de entrada.

Os AG's são normalmente implementados usando simulação num computador, em que um conjunto (designado por população) de representações abstratas (chamadas cromossomas, constituídos por genes) de possíveis soluções (chamados indivíduos) evolui no sentido de descobrir melhores soluções.

Assim, retiram-se algumas grandes diferenças entre os AG's e outras técnicas de resolução de problemas, tais como:

- Não trabalharem diretamente com o domínio do problema mas com representações dos seus elementos;
- Executarem a procura num conjunto de candidatos (população) e não apenas um;
- Não terem conhecimento específico do problema, utilizando apenas a função objetivo;
- Utilizarem basicamente regras probabilísticas.

Deve-se também referir que não é garantida a descoberta da solução ótima! Vão sendo encontradas iterativamente soluções mais adequadas e que vão sendo melhoradas ao longo do algoritmo.

2.2.2 Estrutura e componentes

Nos AG's as variáveis do problema, ou conjunto de parâmetros, são representadas como genes num cromossoma (cromossoma esse que se designa por indivíduo). O valor que o gene

pode tomar é denominado alelo, enquanto a posição no cromossoma (de um determinado gene) é denominado locus.

Os algoritmos genéticos utilizam um conjunto de soluções candidatas a que se dá o nome de população. Recorrendo aos operadores genéticos (seleção, cruzamento, mutação entre outros) é possível obter melhores soluções, ou seja, os cromossomas mais aptos são escolhidos.

O desempenho ou aptidão (*fitness*, na terminologia inglesa) de cada indivíduo (cromossoma) é avaliado com base na função objetivo. A seleção natural permite que os indivíduos mais aptos sejam os progenitores da geração seguinte, ou seja, geram descendentes na população seguinte.

De seguida é possível observar o fluxograma de funcionamento de um algoritmo genético.

Figura 9 – Estrutura de um algoritmo genético

Como se pode verificar, o algoritmo começa por gerar uma população inicial, repetindo depois várias ações até alcançar um determinado número de gerações e atingir uma possível solução.

Em cada geração é calculado o *fitness* de cada indivíduo, com base na função objetivo, para todos os indivíduos que constituem a população. Posteriormente são selecionados, utilizando um dos operadores de seleção, os indivíduos que serão utilizados para a reprodução. Aplicam-

se depois os operadores cruzamento e mutação aos indivíduos selecionados. Os indivíduos obtidos constituem a população da geração seguinte.

O operador cruzamento permite que os genes de dois cromossomas progenitores, previamente selecionados, sejam combinados para formar dois novos cromossomas na população seguinte, os quais, em princípio, serão mais aptos que os seus progenitores, melhorando a população.

São diversas as técnicas de cruzamento existentes que se podem aplicar.

O operador mutação permite introduzir diversidade genética na população. Alterando arbitrariamente um ou mais componentes de uma estrutura escolhida entre a descendência, este operador permite introduzir novos elementos à população, de forma a assegurar que a probabilidade de se chegar a qualquer ponto do espaço de procura não seja nula. Este operador aplica-se geralmente a seguir ao cruzamento. A figura 10 ilustra exemplos das várias técnicas de cruzamento e mutação.

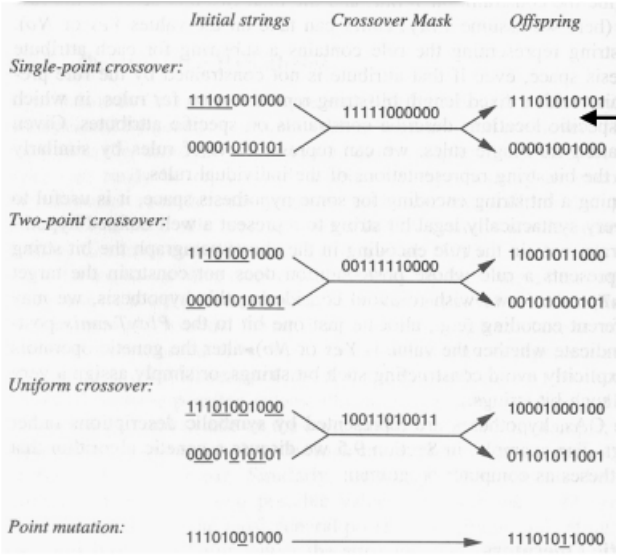


Figura 10 – Técnicas de cruzamento e mutação

A figura 11 faz a síntese dos processos envolvidos na aplicação de um algoritmo genético.

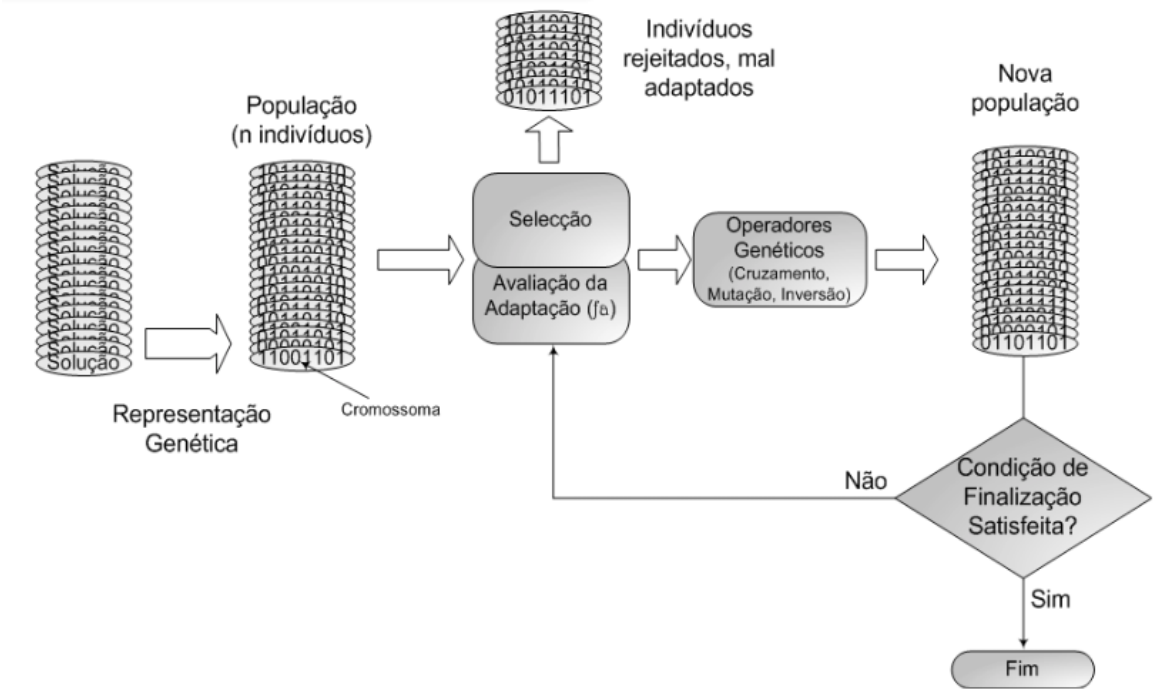


Figura 11 – Esquema funcional de um AG

3. Abordagem ao problema

Depois de evidenciado o problema e conceitos chave, é agora altura de definir caminhos a percorrer para tentar obter uma boa solução. O primeiro passo é investigar o que pode existir já feito e contactar alguns professores das áreas de matemática e informática para ver que soluções existem no sentido de agilizar todo o processo de cálculo.

Problemas de exploração combinatória são de difícil resolução e começam a surgir termos como programação linear, método simplex, entre outros, como possíveis soluções do problema.

Para não se perder demasiado tempo em desenvolver uma interface para input e output de dados, e como o modo consola não era uma solução viável, a primeira abordagem recaiu na utilização do Excel, servindo a folha de cálculo para introdução e apresentação de valores e a linguagem de programação VBA para a implementação do algoritmo.

Depois de alguns testes e abordagens ao problema nesta plataforma e face aos resultados obtidos começarem a consumir demasiado tempo (alguns minutos), houve uma mudança de estratégia. A solução passa assim a assentar numa plataforma Web, fazendo uso de linguagem HTML e CSS para introdução e apresentação de dados e PHP como linguagem de programação para desenvolver o algoritmo que vai solucionar o problema. Desta forma a solução também é mais fácil de distribuir e atualizar. De salientar que a principal diferença é abandonar o método exploração combinatória para tentar obter a melhor solução e partir para a implementação de meta-heurísticas, nomeadamente a utilização de algoritmos genéticos, para encontrar rapidamente boas soluções.

3.1 Método 1 – Microsoft Excel (VBA)

Em termos de interface, o utilizador apenas tem de inserir os valores pretendidos (Quantidade e tamanho) numa lista contínua, selecionar essa mesma lista e pressionar o botão “Calcular”, como se pode observar pela figura 12.

2	Comprimento perfil		650	cm																		
3	Desperdício máximo		30	cm																		
5					250	180	100	90														
6	Listagem de material a cortar																					
7	Peças distintas	4			250	250	250	180	180	180	180	100	100	90	90							
8	Total peças	11																				
9		Qt.	Tamanho																			
10		3	250																			
11		4	180																			
12		2	100																			
13		2	90																			

Calcular

Figura 12 – Método 1 – Folha em Excel

Depois a primeira abordagem foi utilizar os dados da lista selecionada e criar um array com todas as peças a serem cortadas. Basta percorrer os vários elementos da lista selecionada, sabendo que uma coluna representa a quantidade e a outra o tamanho, e ir adicionando desta forma o número correto de vezes cada um dos tamanhos ao array.

Para o exemplo apresentado, o array fica com os seguintes elementos:

	250	250	250	180	180	180	180	100	100	90	90

De seguida, usando estes dados, é necessário obter todas as combinações possíveis. É aqui que começam a surgir os problemas de performance e a lista de peças ainda é bastante curta. Então, numa tentativa de otimizar este processo, é necessário identificar apenas as peças com tamanhos distintos e verificar se a combinação entre elas, retorna uma solução aceitável.

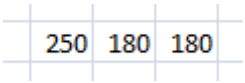
O novo array passa assim apenas a conter os seguintes elementos:

	250	180	100	90

Combinar apenas estes elementos é realmente muito mais rápido face à combinação de todos eles. No entanto, importa perceber se desta combinação resulta um bom plano de corte.

Uma solução possível seria: 250 + 180 + 100 + 90 onde restam 30 (valor dentro do desperdício máximo). Desta forma podem ser logo retiradas todas as peças que permitem

igual plano de corte, deixando as restantes num array para posterior análise. Atendendo a esta consideração podemos remover 2 conjuntos de peças com igual plano de corte, deixando o array apenas com as seguintes peças:



Por sua vez, as peças restantes cabem todas num novo perfil e está assim solucionado o problema. Visto desta maneira, até parece simples, mas na realidade não é, pois se utilizar outras combinações sobre o array inicial, que contém todas as peças, é possível obter um melhor plano de corte, por exemplo: 180 + 180 + 100 + 100 + 90, conseguindo um aproveitamento pleno de um perfil.

No capítulo seguinte será explicado o porquê do abandono desta solução, com mais detalhe.

3.2 Método 2 – Plataforma WEB (PHP)

A adoção deste método tem dois motivos fundamentais:

- A necessidade de poder distribuir rapidamente todas as atualizações que forem surgindo;
- O domínio da linguagem PHP permite maior agilidade no desenvolvimento.

Neste caso o PHP foi escolhido apenas por existir um domínio significativo sobre essa linguagem, mas a solução facilmente poderia ser transposta para qualquer outra.

No capítulo 2 foi abordado o conceito de algoritmo genético. Este é um dos vários métodos que se utilizam para a resolução de problemas complexos. Este método tem por base um processo iterativo sobre uma determinada população fixa, denominados indivíduos, que representam as várias soluções do problema. Esta técnica advém do processo de evolução dos seres vivos demonstrada por Darwin.

Da mesma forma que os sistemas biológicos, ao longo da sua evolução, tiveram que se “moldar” às alterações ambientais para a sua sobrevivência, os algoritmos genéticos acumulam a informação sobre o ambiente com o intuito de se adaptarem ao novo meio. Tal informação funciona como um sistema de triagem para a obtenção de novas soluções exequíveis.

O método dos algoritmos genéticos é muito utilizado devido à simplicidade de operação, eficácia pela determinação da função objetivo e aplicabilidade em problemas onde se

desconhece o modelo matemático ou onde o mesmo se torna impreciso em funções lineares e não-lineares [Costa, 2003].

De forma um pouco sintética esta abordagem passa por, de forma aleatória gerar um conjunto de soluções e depois avaliar a qualidade dessas soluções em função do objetivo, que neste caso é minimizar o desperdício de um dado perfil, tipicamente com um comprimento de 650cm. Basicamente a técnica persiste em preservar as boas soluções a cada iteração, apesar de terem sido geradas à sorte, e descartar aquelas que não interessam.

O processo inicia com a introdução de dados, onde o utilizador, à semelhança do método anterior, deve indicar a quantidade e tamanho das várias peças a cortar. Posteriormente o algoritmo "tenta a sua sorte" na geração de combinações aleatórias, para encontrar uma boa solução. Como se trata de um método de aproximação à solução ótima, não vão ser testadas todas as combinações conforme estava a ser feito no método 1 (3.1). Ao dar a instrução de execução é construído um array com todas as peças. Tomando o mesmo exemplo do método anterior (ver 3.1), nesta fase existe um array com a seguinte informação:

```
Array ( [0] => 250 [1] => 250 [2] => 250 [3] => 180 [4] => 180 [5] => 180 [6] => 180 [7] => 100 [8] => 100 [9] => 90 [10] => 90 )
```

Supondo que o algoritmo visa uma população inicial de 10 combinações, cada vez que o utilizador der ordem de cálculo, vão sendo calculadas possíveis soluções, com a particularidade que à partida vão diferir umas das outras. O algoritmo faz um controlo de soluções válidas, isto é, garante que o somatório das peças não ultrapassa o tamanho do perfil.

Na figura 13 são apresentadas duas soluções geradas pelo algoritmo, onde podemos confirmar a distinção entre ambas.

```
Array
(
    [0] => 250+100+90
    [1] => 180+180+180+90
    [2] => 180+180+90
    [3] => 250+100+100+90+90
    [4] => 250+180+180
    [5] => 250+180+100+90
    [6] => 250+180+100+90
    [7] => 250+250
    [8] => 250+180+100+90
    [9] => 250+180+90
)
```

```
Array
(
    [0] => 250+180+100+90
    [1] => 250+180+180
    [2] => 250+250+90
    [3] => 250+180+100+100
    [4] => 250+180+100
    [5] => 180+180+180+90
    [6] => 250+180+90
    [7] => 180+180+90
    [8] => 180+180+100
    [9] => 250+180
)
```

Figura 13 – Exemplo de soluções geradas pelo algoritmo

O array que possui todas as soluções é ordenado em função do somatório das suas peças e da diferença que existe para com o tamanho do perfil, ficando no topo a solução que minimiza o desperdício, ou seja, tem menor sobra. A posição de topo é assim a melhor das soluções geradas para um plano de corte.

A partir daqui, num processo iterativo, as peças usadas no plano de corte eleito, são removidas do array inicial que contém todas as peças a serem cortadas e todo o processo se repete até que não haja mais peças para cortar.

Para cada iteração é possível observar as várias soluções calculadas. A figura 14 demonstra esse cenário:

Perfil (barra) nº 1

	Plano de corte	Somatório	Sobra
000	180+180+100+100+90	650	0
001	250+180+100+90	620	30
002	250+180+180	610	40
003	250+180+90+90	610	40
004	180+180+90+90	540	110
005	250+180+100	530	120
006	250+180+90	520	130
007	250+180+90	520	130
008	250+250	500	150
009	180+100+90+90	460	190

Perfil (barra) nº 2

	Plano de corte	Somatório	Sobra
000	250+250+90	590	60
001	250+180+90	520	130
002	250+180+90	520	130
003	250+180+90	520	130
004	250+180+90	520	130
005	250+250	500	150
006	180+180+90	450	200
007	180+180+90	450	200
008	180+180+90	450	200
009	250+180	430	220

Perfil (barra) nº 3

	Plano de corte	Somatório	Sobra
000	250+180+180	610	40
001	250+180+180	610	40
002	250+180+180	610	40
003	250+180+180	610	40
004	250+180+180	610	40
005	250+180+180	610	40
006	250+180+180	610	40
007	250+180+180	610	40
008	250+180+180	610	40
009	250+180+180	610	40

Figura 14 – Soluções a cada iteração

No final é apresentado o plano de corte que corresponde à melhor solução encontrada para cada iteração. Todo este processo é executado em menos de 3 milésimos de segundo.

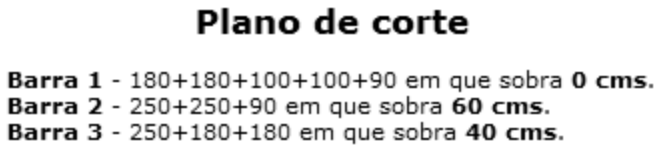


Figura 15 – Plano de corte final

4. Codificação e implementação

Nesta secção são explicadas como foram implementadas as ideias apresentadas no capítulo anterior, com mais detalhe. Vão ser apresentadas as abordagens ao problema e respetivas conclusões à medida que foram tomadas decisões.

4.1 Método 1 – Microsoft Excel (VBA)

Como já foi descrito no capítulo anterior o maior problema desta abordagem recai sobre o algoritmo responsável por efetuar todas as combinações possíveis a partir do array que contém todas as peças a serem cortadas.

Tomando o mesmo exemplo do capítulo anterior, se apenas tiver em consideração as peças com tamanhos distintos (250, 180, 100 e 90), o algoritmo vai apresentar o resultado apresentado na figura 16.

	A	B	C
1	**Combinações**	**Total**	**Sobra**
2	250	250	400
3	180	180	470
4	100	100	550
5	90	90	560
6	250 + 180	430	220
7	250 + 100	350	300
8	250 + 90	340	310
9	180 + 100	280	370
10	180 + 90	270	380
11	100 + 90	190	460
12	250 + 180 + 100	530	120
13	250 + 180 + 90	520	130
14	250 + 100 + 90	440	210
15	180 + 100 + 90	370	280
16	250 + 180 + 100 + 90	620	30

Figura 16 – Cálculo de combinações

Como é possível observar, são testadas todas as combinações possíveis para os dados fornecidos. Todo este cálculo, demora cerca de 1 segundo. Se usar todas as peças a serem cortadas é preciso esperar aproximadamente 90 segundos[2] para obter uma listagem de resultados que apresenta todas as combinações possíveis. Esta listagem é um pouco extensa e foram utilizadas 2048 linhas para apresentar os resultados. Na figura 17 é possível observar algumas partes dessa listagem, escolhidas de forma aleatória.

[2] Tempo medido num computador com as características que constam no capítulo seguinte.

	A	B	C
1	**Combinações**	**Total**	**Sobra**
2	250	250	400
3	250	250	400
4	250	250	400
5	180	180	470
61	180 + 90	270	380
62	100 + 100	200	450
63	100 + 90	190	460
75	250 + 250 + 90	590	60
77	250 + 250 + 180	680	-30
88	250 + 180 + 100	530	120
315	250 + 180 + 100 + 90	620	30
317	250 + 180 + 90 + 90	610	40
318	250 + 180 + 180 + 180	790	-140
492	250 + 100 + 90 + 90	530	120
493	180 + 180 + 180 + 180	720	-70
1018	180 + 100 + 100 + 90 + 90	560	90
1019	180 + 180 + 100 + 100 + 90	650	0
1021	180 + 180 + 100 + 90 + 90	640	10
1025	250 + 250 + 250 + 180 + 180 + 180	1290	-640
1481	180 + 180 + 180 + 100 + 100 + 90	830	-180
1482	180 + 180 + 180 + 100 + 90 + 90	820	-170
1484	180 + 180 + 100 + 100 + 90 + 90	740	-90
1805	250 + 180 + 180 + 180 + 100 + 90 + 90	1070	-420
1808	250 + 180 + 180 + 100 + 100 + 90 + 90	990	-340
1809	180 + 180 + 180 + 180 + 100 + 100 + 90	1010	-360
1816	180 + 180 + 180 + 100 + 100 + 90 + 90	920	-270
1817	250 + 250 + 250 + 180 + 180 + 180 + 180 + 100	1570	-920
2036	250 + 180 + 180 + 180 + 180 + 100 + 100 + 90 + 90	1350	-700
2037	250 + 250 + 250 + 180 + 180 + 180 + 180 + 100 + 100 + 90	1760	-1110
2044	250 + 250 + 250 + 180 + 180 + 180 + 100 + 100 + 90 + 90	1670	-1020
2045	250 + 250 + 180 + 180 + 180 + 180 + 100 + 100 + 90 + 90	1600	-950
2048	250 + 250 + 250 + 180 + 180 + 180 + 180 + 100 + 100 + 90 + 90	1850	-1200

Figura 17 – Listagem com cálculo de combinações

Esta listagem depois de ordenada pela sobra, permite identificar bons planos de corte. A figura 18 mostra precisamente esses planos. As linhas ocultas entre os vários resultados representam soluções iguais.

	A	B	C
1	**Combinações**	**Total**	**Sobra**
2	180 + 180 + 100 + 100 + 90	650	0
14	180 + 180 + 180 + 100	640	10
22	180 + 180 + 100 + 90 + 90	640	10
34	250 + 180 + 100 + 100	630	20
46	180 + 180 + 180 + 90	630	20
54	250 + 100 + 100 + 90 + 90	630	20
57	250 + 180 + 100 + 90	620	30
105	250 + 180 + 180	610	40
123	250 + 180 + 90 + 90	610	40
135	250 + 250 + 100	600	50
141	250 + 250 + 90	590	60
147	180 + 180 + 100 + 100	560	90

Figura 18 – Melhores planos de corte

Ainda muito poderia ser feito para melhorar este método, como por exemplo abortar o ciclo sempre que a combinação excede a medida máxima do perfil. Esta alteração elimina logo, neste caso, 1646 linhas de resultado. No entanto, depois é sempre necessário repetir todos estes passos para as peças que vão sobrando, à medida que se retiram aquelas que constituem um bom plano de corte.

Certamente que esta abordagem muito provavelmente levaria a resultados ótimos, dado ser feita uma comparação exaustiva entre todas as possibilidades. Dada a complexidade do problema, resta saber quanto tempo seria necessário esperar para obter a solução, dependendo obviamente da quantidade de peças que se pretende cortar. O exemplo apresenta uma listagem de peças tipicamente curta no mundo real e o tempo de espera só para apresentar os dados até agora expostos, ultrapassa já os 90 segundos.

Com tais factos, é altura de mudar a abordagem e partir para outra solução, na busca de boas soluções com tempos de resposta preferencialmente curtos, tipicamente inferiores a 5 segundos.

4.2 Método 2 – Plataforma WEB (PHP)

Continuando agora, com mais detalhe, aquilo que foi introduzido no capítulo anterior referente a este método, o desafio começa logo pela necessidade de criar uma interface onde o utilizador possa indicar as peças a cortar e algumas configurações como o tamanho do perfil e número de tentativas que o algoritmo deve efetuar.

A solução passa assim por construir uma interface bastante simples e intuitiva para que seja usável com extrema facilidade por parte dos operários que vão fazer uso dela. A figura 19 mostra a solução implementada:

Figura 19 – Interface inicial

O utilizador dispõe de uma caixa de listagem onde, respeitando a sintaxe definida (uma linha deve expressar a quantidade e o tamanho único), deve introduzir a listagem de peças a cortar. Pode ainda definir o tamanho do perfil (barra). O desperdício vai servir apenas para uma questão visual, fazendo realçar que parte do material é considerado lixo.

Este método foi sofrendo várias melhorias e alterações à medida que se avançava quer no desenvolvimento, quer nos testes com a empresa que reportou esta necessidade e que de algum modo contribui para atingir o estado final, em termos de algoritmo, a que a solução chegou.

Numa primeira abordagem tudo é processado conforme descrito até aqui. O resultado final é o apresentado na figura 20. Do lado esquerdo é mantida a informação introduzida pelo utilizador, do lado direito são apresentados os sucessivos cálculos que são efetuados a cada iteração e no final o utilizador fica a saber qual o melhor plano de corte para a listagem fornecida. Para efeitos de controlo de desempenho é apresentado o tempo que o algoritmo utilizou para processar toda a informação.

Figura 20 – Aspeto do resultado final

Esta abordagem era já motivo de grande satisfação quando comparado com o método 1, quer nas soluções obtidas quer no contraste que existe em termos de tempos de execução. Um decréscimo de 90s para cerca de 5ms é uma diferença abismal, perante o mesmo problema.

Ainda assim havia a necessidade de analisar o comportamento do algoritmo com um volume de dados consideravelmente maior.

Não demorou a perceber que se passar de 10 a 20 peças para valores muito superiores como 1.000, 5.000 ou mesmo 10.000 as coisas variam, e muito, no que toca ao desempenho.

Um pequeno teste, consistiu em aumentar 100 vezes a dimensão do problema, passando para um total de 1100 peças a cortar, que influência significativamente o tempo de processamento necessário. Se antes em cerca de 5ms se obtinha o resultado, desta vez já são necessários cerca de 1,1s.

4.3 Melhorias de interface e funcionalidade

Das diversas trocas de opinião tidas com a empresa, foram notadas algumas necessidades no sentido de melhorar a interface de modo a permitir o seguinte:

- Como existem sobras de outros cortes, seria de todo vantajoso poder definir o tamanho para cada perfil de material e não assumir que todos têm o mesmo valor;
- Uma porta ou janela habitualmente usam mais que um perfil de alumínio na sua constituição, logo deve ser possível diferenciar os perfis e as peças a cortar em cada um desses perfis;
- Poder imprimir o resultado, com os vários planos de corte, num formato mais agradável que imprimir a página Web;

Atendendo a todas estas considerações, e sempre com o objetivo de manter simples a interface para a introdução de dados, a mesma foi reestruturada e deve agora obedecer a outra sintaxe, para permitir assim distinguir os diversos perfis, bem como definir o comprimento de cada um deles, no que toca à introdução de dados. A apresentação da página foi também reestruturada para apresentar uma imagem mais profissional.

A figura 21 apresenta a primeira abordagem**.** O utilizador apenas tinha de indicar a lista de material a cortar. Todos os perfis tinham assim o mesmo tamanho.

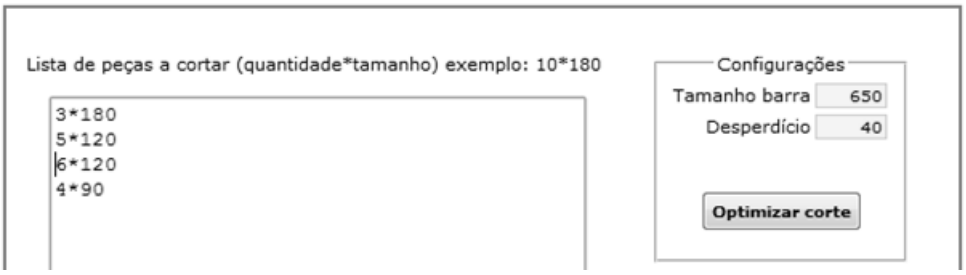


Figura 21 – Aspeto inicial da interface

A figura 22 mostra a nova interface. Passa a existir uma linha que identifica o perfil e comprimento. Só de seguida são introduzidas as peças a cortar nesse perfil. A sintaxe é rígida e deve ser respeitada, no entanto o utilizador tem sempre disponível ajuda em contexto, acompanhada de um exemplo.

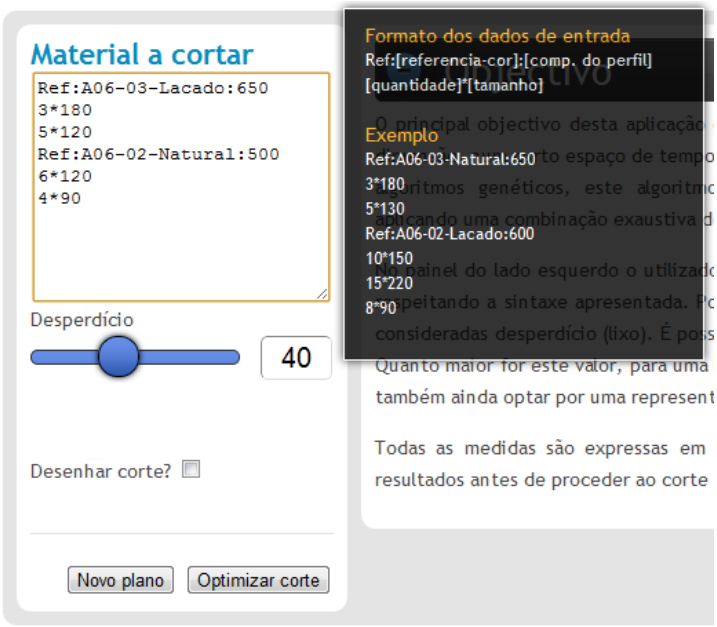


Figura 22 – Aspeto final da interface

Na primeira linha a palavra “Ref” serve para determinar que se trata de uma linha onde vão ser definidas as características do perfil e de seguida terá a lista de peças até aparecer nova linha com a instrução “Ref”. A separação entre valores é feita com o símbolo “:”. Deve indicar uma referência (texto que designe o perfil) seguido da cor, pois existem perfis iguais mas de cor diferente, seguindo-se o tamanho expresso em centímetros.

Para evitar possíveis erros foram alterados os controlos que permitem indicar o desperdício. O utilizador pode ainda optar por visualizar uma representação gráfica da solução, que acompanha a solução textual.

Esta nova interface permite de uma só vez, otimizar os cortes para perfis de comprimentos distintos e apresentar logo todos os resultados, sendo muito mais útil em termos de utilização real, como se pode observar na figura 23.

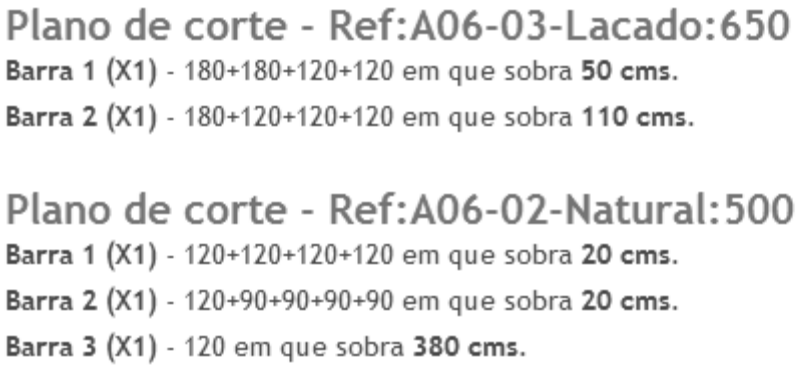


Figura 23 – Resultado para várias referências de material

É possível observar agora que, para cada plano de corte, aparece o número de vezes que esse mesmo plano deve ser cortado. No exemplo apresentado, cada plano só pode ser aplicado uma única vez, visto não existirem peças que possibilitem aplicar o mesmo plano várias vezes mantendo a otimização de material. Alterando o número de peças, já é possível obter várias barras de material com igual plano de corte, como se pode observar na figura 24.

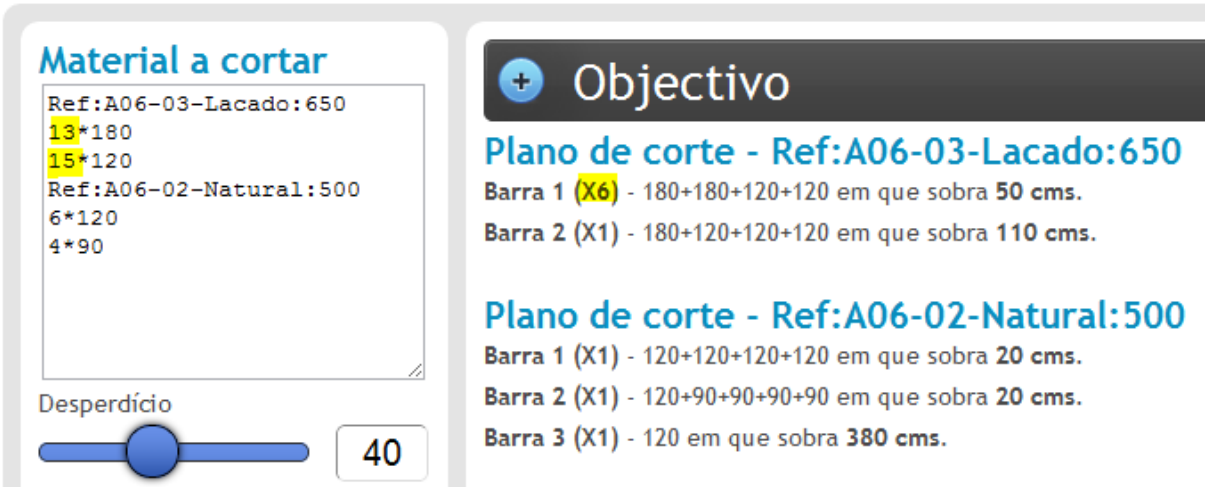


Figura 24 – Barras com mesmo plano de corte

Se o utilizador optar por ativar a opção para desenhar o corte, vai obter o resultado da figura 25.

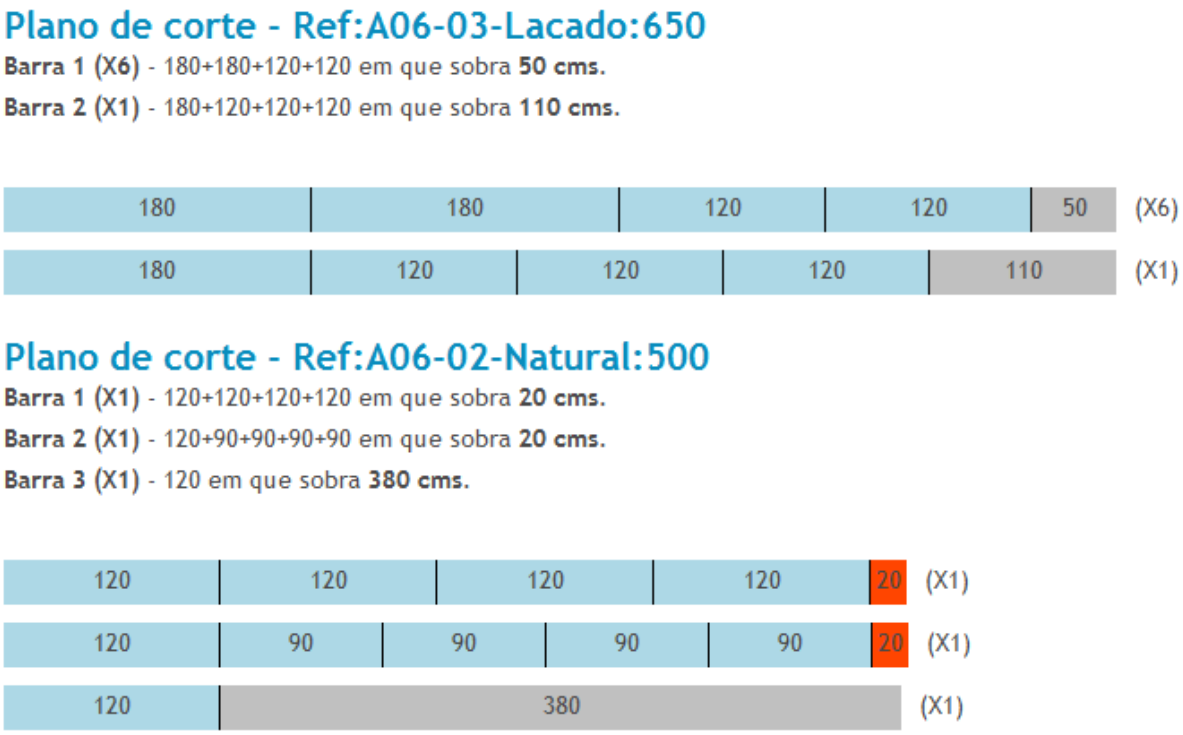


Figura 25 – Desenhar plano de corte

Só com esta informação visível podemos observar pela primeira vez o objetivo do parâmetro "desperdício", que assinala todas as partes com medida inferior à indicada a uma cor diferente, o que significa que é considerado lixo.

Em termos de oficina é mais prático olhar para o desenho dos cortes do que para a informação textual. Houve também a preocupação de manter à escala todos os valores, independentemente do tamanho indicado para cada perfil, para que toda a informação seja apresentada dentro do layout previsto.

Em termos de impressão, também foi tido em conta o desperdício de papel e por isso toda a informação adicional irrelevante foi removida. A largura da representação das barras também foi diminuída. A figura 26 mostra uma pré-visualização da impressão, para o exemplo em uso.

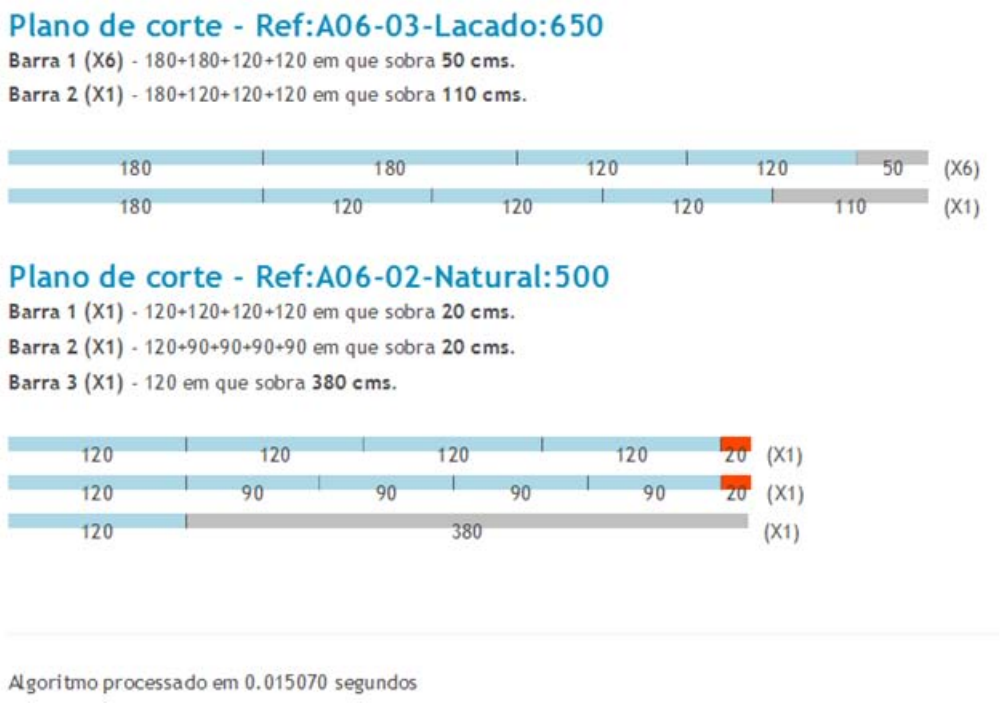


Figura 26 – Pré-visualização da impressão

Como já foi dito anteriormente toda a interface foi revista, depois de o algoritmo ter dado já provas de bom funcionamento. Logo de início é apresentado um breve texto explicativo sobre o objetivo da aplicação, para que novos utilizadores possam ficar a perceber a sua utilidade e funcionamento. A usabilidade da aplicação foi também revista e agora mais adaptada aos utilizadores finais. A ajuda em contexto, aparece sempre que o utilizador se posiciona na caixa onde deve introduzir o material a cortar.

O "*slider*" aplicado ao desperdício permite balizar este parâmetro entre valores razoáveis e pré-definidos. Neste caso o parâmetro desperdício acaba por não ter praticamente qualquer influência, pois apenas é utilizado na representação gráfica do plano de corte, sendo o seu impacto na performance muito diminuto.

Sempre que seja dada ordem de otimização o texto do objetivo fica minimizado, dando lugar aos resultados. Porém, a qualquer instante pode aceder novamente ao texto, clicando sobre "Objetivo". A figura 27 mostra o ecrã inicial da aplicação.

Figura 27 – Ecrã inicial da aplicação final

4.4 Duplicar plano de corte

A figura 28, que contém parte dos resultados, permite constatar que são identificadas diversas vezes a mesma solução, apresentando um plano de corte igual, que pode ser aplicado enquanto existirem peças suficientes a cortar.

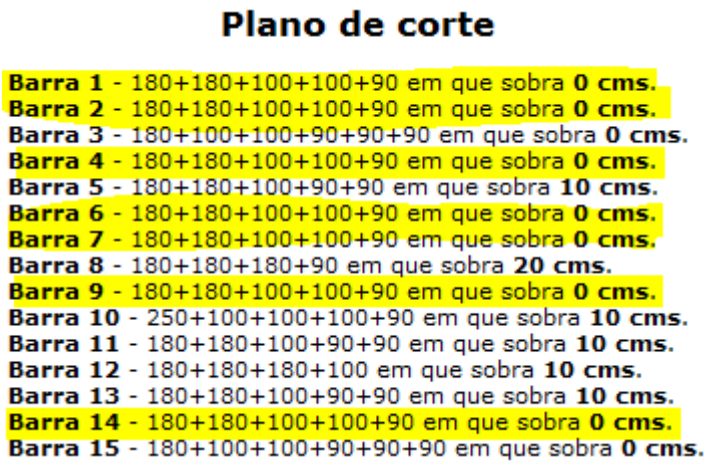


Figura 28 – Planos de corte iguais

Assim, como já havia sido previsto para o método 1, mas não implementado, era oportuno verificar se compensa, uma vez encontrado um plano de corte, repetir este mesmo plano enquanto existirem peças que o permitam aplicar, ou se por sua vez é preferível deixar correr o algoritmo naturalmente. O algoritmo foi assim modificado para remover todas as peças a cortar que permitam formar um plano de corte igual ao selecionado. Finalizado este processo o algoritmo procede de igual forma com as peças que ainda existem para cortar. Pode-se dizer que os ganhos com esta alteração são muito substanciais e no capítulo seguinte serão vistos os resultados com mais detalhe.

4.5 Eliminar soluções duplicadas

Diretamente relacionado com a quantidade de peças distintas a cortar, estão as diversas combinações possíveis de realizar.

A situação aqui relatada visa demonstrar que se tivermos muitas peças a cortar, mas de apenas 2 ou 3 tamanhos distintos, por mais tentativas que se executem o algoritmo vai acabar por gerar na população inicial soluções com muitas combinações repetidas. Desta forma a melhoria aqui prevista é, depois de gerada a população inicial de dados (combinações de corte possíveis), remover todas as soluções duplicadas e aproveitar apenas as soluções distintas para prosseguir com a execução.

Nas figuras 29 e 30 é demonstrada a situação referida. Mesmo que a “População Inicial” fosse maior é pouco provável, senão mesmo impossível para os dados em questão, que se consigam obter mais combinações distintas do que aquelas apresentadas como “População Única”.

Figura 29 – Listagem de material a cortar

Populacao Inicial:

```
Array
(
    [0] => 180+180+120+120
    [1] => 180+120+120+120
    [2] => 180+180+180
    [3] => 180+180+120+120
    [4] => 180+120+120+120
    [5] => 180+180+120+120
    [6] => 180+180+180
    [7] => 180+180+120+120
    [8] => 180+180+120+120
    [9] => 180+180+120+120
    [10] => 180+180+120+120
    [11] => 180+180+120+120
    [12] => 180+180+180
    [13] => 180+180+120+120
    [14] => 180+180+120+120
    [15] => 180+180+180
    [16] => 180+180+120
    [17] => 180+120+120+120
    [18] => 180+180+120+120
    [19] => 180+180+180
)
```

Populacao Única:

```
Array
(
    [0] => 180+120+120+120
    [1] => 180+180+120+120
    [2] => 180+180+180
    [3] => 180+180+120
    [4] => 120+120+120+120+120
)
```

Figura 30 – Combinações calculadas e combinações únicas

4.6 Otimizar em função do desperdício

No decorrer de várias experiências, surgiu uma situação que importa aqui reportar e perceber até que ponto a melhor solução é efetivamente aquela que minimiza o desperdício de material, ou se existem situações em que se deve ter uma abordagem diferente e permitir desperdícios maiores, sendo assim considerados restos úteis para obras seguintes.

Nas figuras seguintes (31, 32 e 33) é possível observar o problema. Ao acaso, para a listagem de material em questão, onde existe pouca diversidade de peças mas muita quantidade, o algoritmo pode apresentar qualquer uma das duas soluções:

Nesta primeira situação o algoritmo apresenta uma solução sempre com o mesmo plano de corte, deixando uma sobra de 50cm.

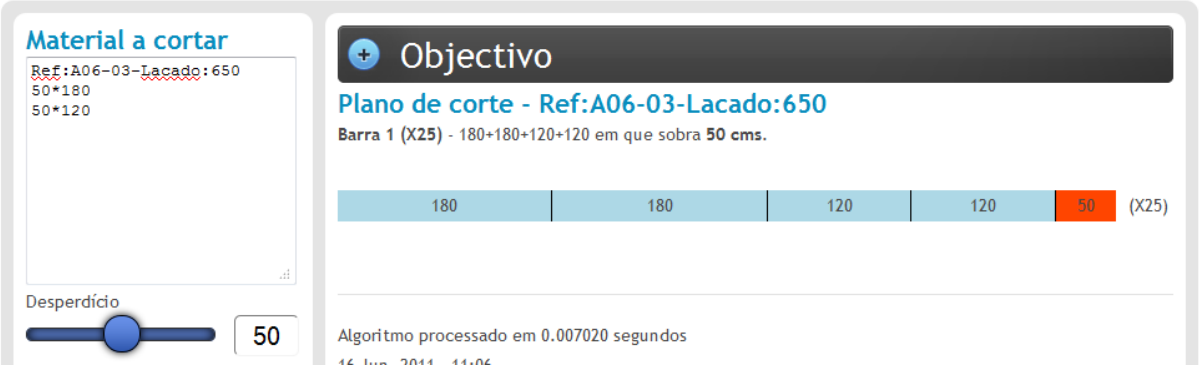


Figura 31 – Um resultado possível

Na situação seguinte o algoritmo apresenta outra solução que contempla 3 planos de corte distintos.

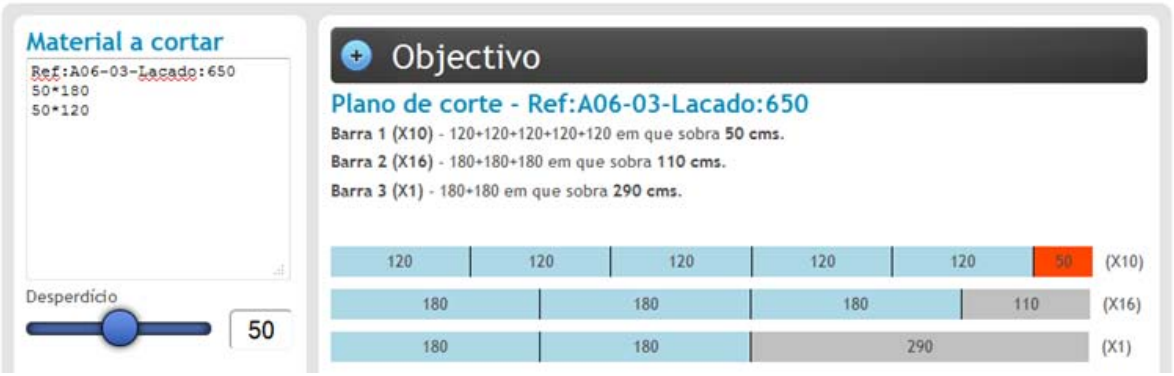


Figura 32 – Outro resultado possível, para o mesmo material

O dilema agora é saber qual é efetivamente a melhor das soluções. A situação foi discutida com o responsável da empresa, que justificou não haver uma fórmula que possa ser aplicada, pois depende do material em uso.

Para ajudar a compreender esta abordagem, aqui fica um exemplo.

Se um determinado cliente com exigências diferente do normal solicitar por exemplo um perfil de alumínio numa cor a gosto diferente das tradicionais, a melhor escolha é aquela que usa o menor número de barras, pois não interessa ter sobras que possam ser aproveitadas para obras seguintes dada a especificidade do material. No entanto se o material escolhido for mais tradicional, então a solução mais favorável é aquela que permite ter o melhor equilíbrio entre sobras que são consideradas lixo e as sobras que se podem aproveitar para obras seguintes. É preferível ter mais planos de corte em que sobre 110cm ou mais, do que ter várias a sobrar 50cm, que à partida já será considerado lixo. Esta situação usando o 1º plano de corte acaba por desperdiçar 25x50 (1250cm) de material que não é à partida reutilizado.

Para contornar este problema, e dar mais controlo ao utilizador sobre o comportamento do algoritmo, vai ser feito um aproveitamento do valor do desperdício.

Isto é, deixa de ser apenas um valor que tem representação gráfica para os desperdícios (vermelho), mas passa a influenciar o resultado das soluções geradas. Quanto menor for o valor, menos são os desperdícios. Aliás, quando colocado a zero, o algoritmo passa a ter um comportamento semelhante ao de tantas outras soluções que se centram apenas em minimizar o desperdício. Se aproximarmos o valor do desperdício para próximo do seu valor máximo, o algoritmo vai privilegiar soluções cujas sobras tenham comprimento suficiente para serem aproveitadas numa próxima obra. Assim, em vez dos resultados obtidos serem apenas baseado na “sorte” do algoritmo, o utilizador passa a poder controlar mais um parâmetro para poder afinar os resultados e analisar a situação mais favorável a cada caso.

Exemplo de resultados para vários valores de desperdício e igual número de peças a cortar:

A figura 33 apresenta o resultado colocando o valor para o desperdício a zero.

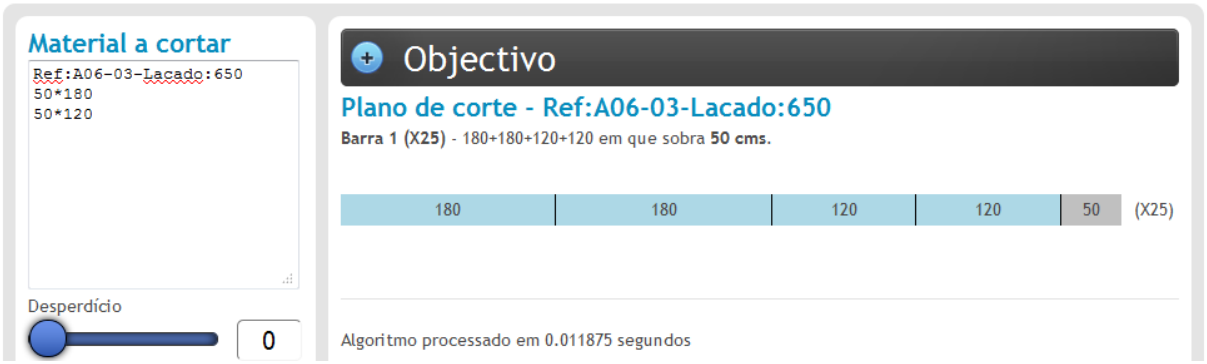


Figura 33 – Desperdício a zero

Na figura 34, é possível verificar o resultado obtido com desperdício a 50.

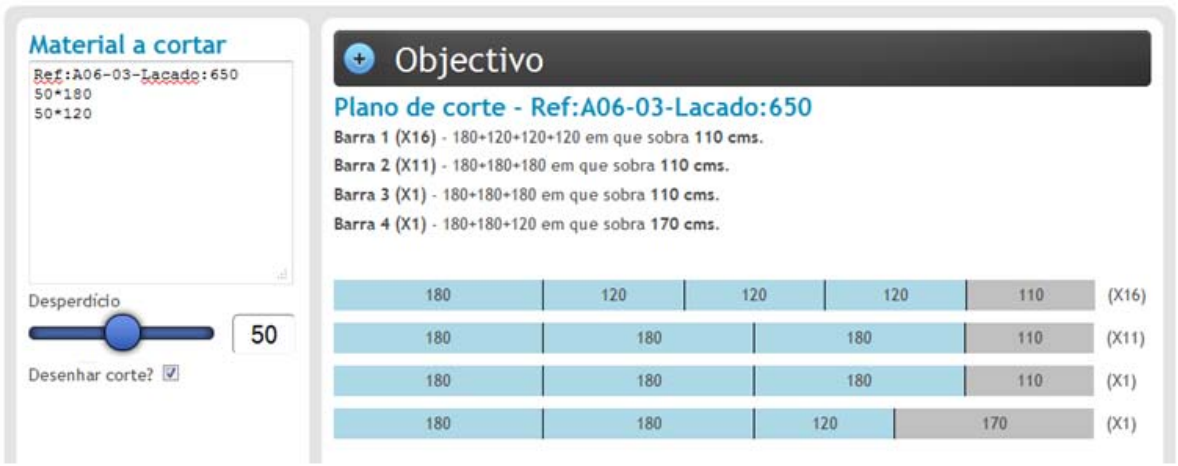


Figura 34 – Desperdício a 50

Para finalizar esta sequência de testes a figura 35 apresenta o resultado com o desperdício a 100.

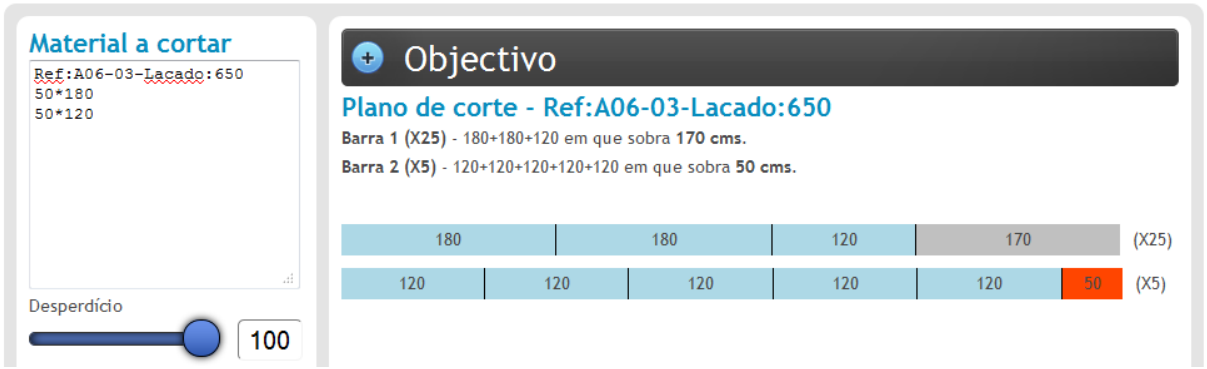


Figura 35 – Desperdício a 100

A partir das 3 imagens anteriores é possível observar que para a mesma listagem de material são produzidos resultados bastante distintos, dando assim alguma liberdade de escolha ao utilizador.

Na sequência desta implementação, surgiu outra necessidade que é explicada no ponto seguinte.

4.7 Guardar e comparar resultados

Visto que cada vez que se executa a aplicação os resultados anteriores são perdidos, houve a necessidade premente de poder guardar os resultados para comparação. Desta forma foi adicionada essa funcionalidade na interface.

Uma vez gerada uma solução é possível guardar o resultado, como se pode observar na figura 36.

Figura 36 – Guardar resultado

Depois de guardado o resultado, este fica disponível para comparação nas execuções seguintes, possibilitando ao utilizador uma melhor visão global sobre as várias soluções calculadas. Juntamente a cada uma das soluções é apresentado um resumo estatístico contendo o total de barras utilizadas e sobras. Para cada solução de corte é possível identificar a percentagem de utilização.

É possível guardar diversos resultados para análise. Também se podem eliminar aqueles que deixam de ter interesse quando comparados com outros que melhor satisfaçam a necessidade do operador.

Esta funcionalidade de guardar resultados, aliada à informação estatística, é deveras interessante e pertinente na medida em que se torna um instrumento extremamente útil no auxílio da tomada de decisão, a partir de informação detalhada sobre cada plano apresentado.

A figura 37 ilustra a situação em que já existe um resultado guardado para comparação, bem como o botão que permite eliminar um resultado guardado.

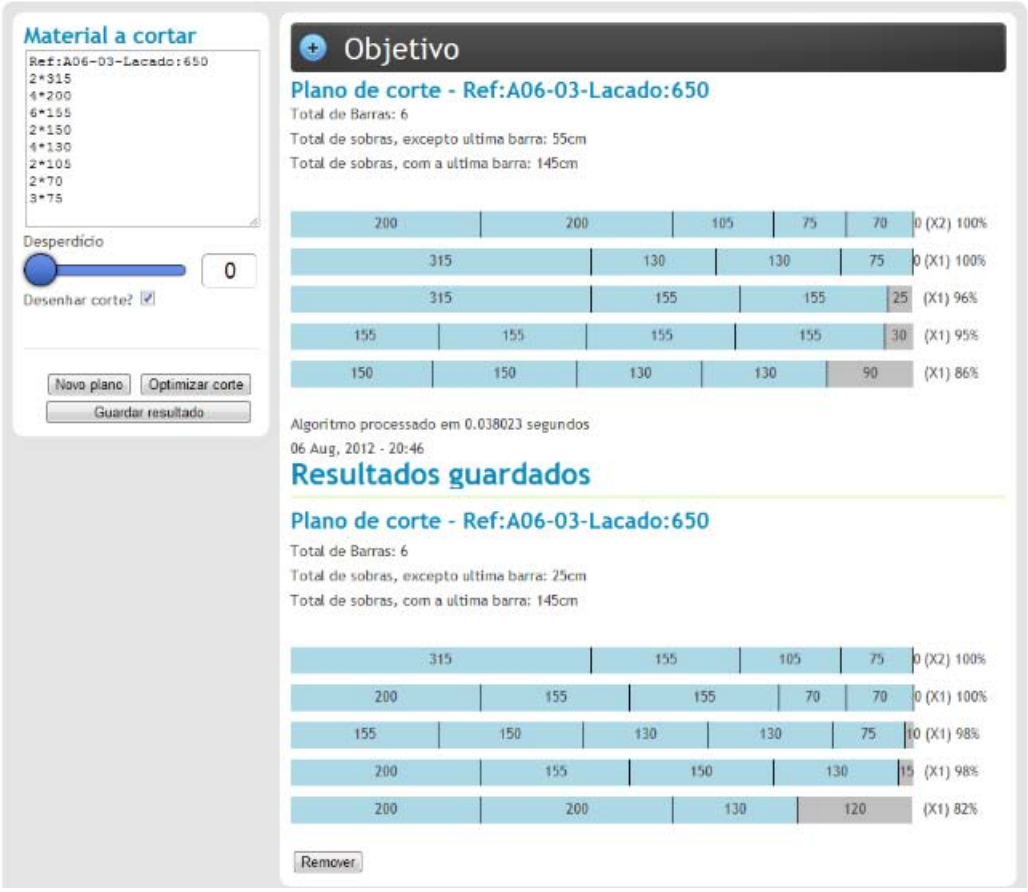


Figura 37 – Comparação de resultados

5. Análise de resultados

Os resultados finais aqui apurados, bem como os tempos referidos no capítulo anterior, foram realizados num computador com as seguintes características:

Processador:	Intel(R) Pentium(R) Dual CPU E2180 @ 2.00GHz 2.24 GHz
Memória instalada (RAM):	3,00 GB (2,87 GB utilizável)
Tipo de sistema:	Sistema Operativo de 64 bits

Alguns testes realizados sobre os Excel não são levados em consideração, dado terem tempos muito elevados para o processamento dos dados e não ter sido a solução adotada para a resolução do problema.

Como cada solução apresentada é sempre uma boa solução, sem qualquer garantia de ser a melhor, sempre que o algoritmo é processado os tempos de execução sofrem sempre ligeiras variações, mesmo mantendo os dados inalterados. Desta forma os dados apresentados no quadro seguinte resultam da média, efetuada a partir de 50 simulações para os mesmos dados, usando 30 tentativas, permitindo assim obter valores mais realistas.

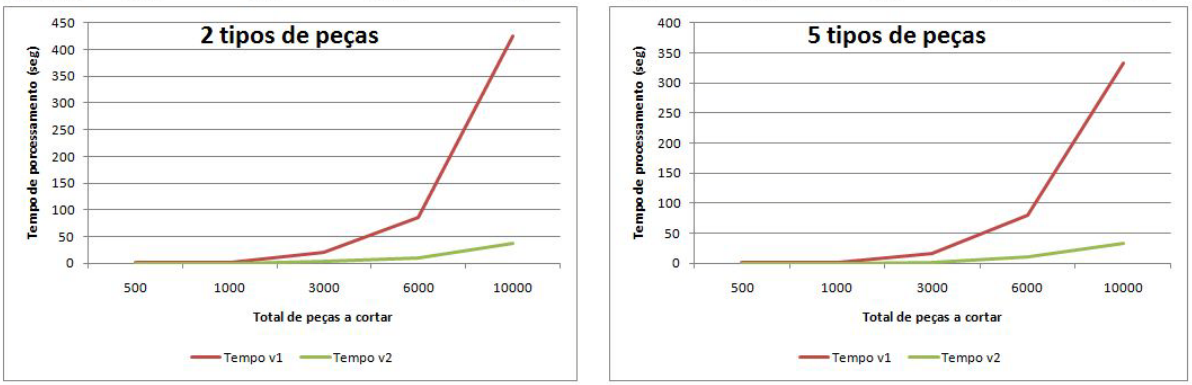
Foram feitos essencialmente dois testes, onde se compara o algoritmo inicial (v1), sem a remoção de planos de corte similares, caso existam, com a segunda versão (v2) onde antes de partir para novo cálculo, são retiradas todas as peças que permitem igual plano de corte. Os testes assentam em diferentes quantidades de peças, tendo um aumento progressivo desde 10 até 10.000. Também foi comparado, para o mesmo número de peças, se a diversidade de medidas afeta de forma significativa os tempos de processamento. A parte superior do quadro

5 usa apenas 2 medidas distintas (180 e 120), enquanto a segunda parte usa 5 (180, 120, 110, 90 e 70). Este fator não se revelou significativo nos resultados apurados.

Quadro 5 – Quadro comparativo de resultados

	QT	Tam	QT	Tam	QT	Tam	QT	Tam	QT	Tam	QT	Tam	QT	Tam	QT	Tam
	5	180	15	180	50	180	100	180	250	180	500	180	1500	180	3000	180	5000	180
	5	120	15	120	50	120	100	120	250	120	500	120	1500	120	3000	120	5000	120
Total Peças	10		30		100		200		500		1000		3000		6000		10000	
Tempo v1	0,007767		0,019223		0,079546		0,198249		0,697484		2,318826		20,8383		85,35607		425,4077	
Tempo v2	0,004656		0,005433		0,014837		0,025821		0,083262		0,30393		2,76831		11,12356		37,76798	
Relação v1/v2	167%		354%		536%		768%		838%		763%		753%		767%		1126%	

	QT	Tam	QT	Tam	QT	Tam	QT	Tam	QT	Tam	QT	Tam	QT	Tam	QT	Tam	QT	Tam
	2	180	6	180	20	180	40	180	100	180	200	180	600	180	1200	180	2000	180
	2	120	6	120	20	120	40	120	100	120	200	120	600	120	1200	120	2000	120
	2	110	6	110	20	110	40	110	100	110	200	110	600	110	1200	110	2000	110
	2	90	6	90	20	90	40	90	100	90	200	90	600	90	1200	90	2000	90
	2	70	6	70	20	70	40	70	100	70	200	70	600	70	1200	70	2000	70
Total Peças	10		30		100		200		500		1000		3000		6000		10000	
Tempo v1	0,005942		0,020676		0,069124		0,178715		0,590898		2,151834		16,57971		80,98219		333,7462	
Tempo v2	0,005672		0,010354		0,017486		0,032523		0,096255		0,301702		2,291354		10,53711		33,23339	
Relação v1/v2	105%		200%		395%		550%		614%		713%		724%		769%		1004%	



O que se pode concluir pela análise da tabela, é uma clara melhoria no desempenho do algoritmo na sua última versão (v2), verificando-se uma proporcionalidade entre o número de peças a cortar e o ganho.

Em todo caso, e de acordo com a opinião recolhida na empresa, não é de todo habitual ter uma obra onde tenham de cortar mais de 500 peças de uma única vez.

Outro passo não menos importante é tentar perceber quais os custos computacionais que cada função interveniente no algoritmo representa no total, para que possíveis esforços de otimização sejam dirigidos à parte que causa mais impacto na performance e desempenho global.

A figura 38 ilustra um extrato dos *logs* criados no sentido de apurar com exatidão os custos de diversas funções.

```
Tentativas; Total peças; Tempo
30;1675;0,70853304862976
Criar estrutura: 0.0002%
     Tempo: 0,00012993812561035
     Execuções: 1
Pick Random: 10.7887%
     Tempo: 0,076441526412964
     Execuções: 210
Rate: 0.2457%
     Tempo: 0,0017411708831787
     Execuções: 85
Remover Duplicados: 0.0439%
     Tempo: 0,00031089782714844
     Execuções: 7
Duplicar Plano de Corte (setNulos e removerNulosArr): 79.5136%
     Tempo: 0,56338000297546
     Execuções: 315
    - Set Nulos: 1.2697%
         Tempo: 0,0071535110473633
         Execuções: 308
    - Remover Nulos Array: 58.1037%
         Tempo: 0,3273446559906
         Execuções: 315
```

Figura 38 – Log de resultado

No exemplo apresentado na figura 38, o cenário contempla um universo de 1675 peças para cortar, onde fica claro que o processo de encontrar planos duplicados, que por sua vez chama outros dois (Set Nulos e Remover Nulos Array), é o responsável por uma percentagem muito significativa atendendo ao total de tempo necessário para obter uma solução.

O gráfico da figura 39 compila todos os dados apurados e permite verificar a percentagem de processamento despendida em função do número de peças a cortar.

Dentro dos testes efetuados é possível verificar que à medida que o número de peças a cortar aumenta, também aumenta o tempo despendido pela função já identificada, atingindo mesmo valores na ordem dos 90%. Fica assim claro que futuros esforços de otimização devem ser focalizados concretamente nesta função específica.

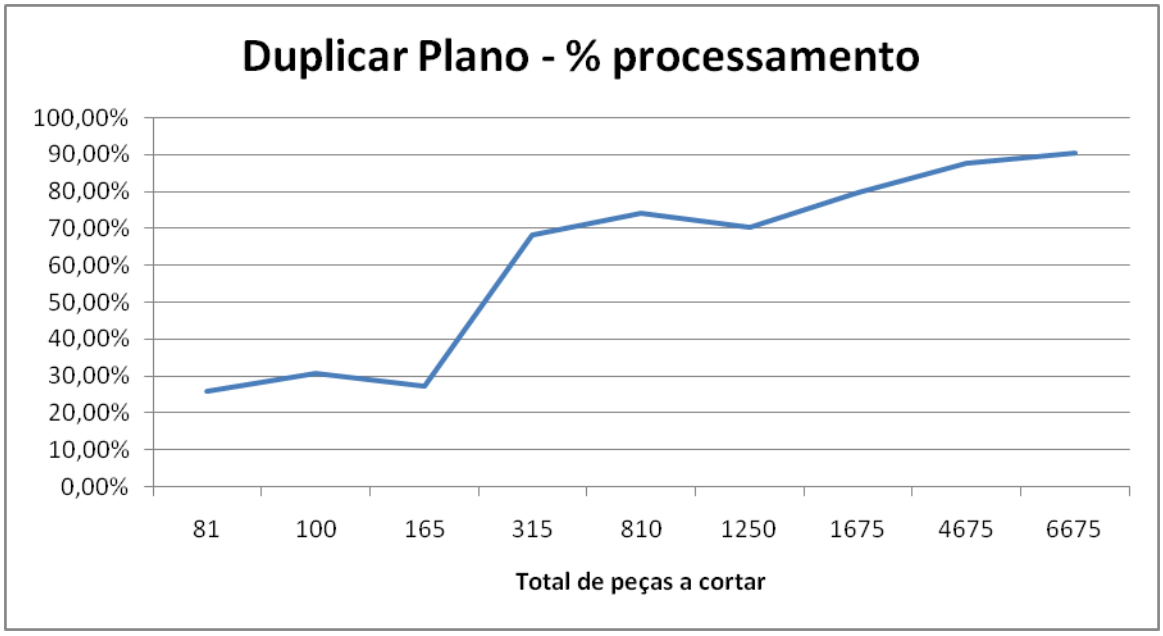


Figura 39 – Percentagem de processamento – duplicar plano

A figura 40 apresenta outro extrato parcial do *log* criado, desta vez com um menor número de peças, onde se pode comprovar os tempos gastos pelos diversos intervenientes em análise.

```
Tentativas; Total peças; Tempo
30;81;0,012629985809326
Criar estrutura: 0.0048%
    Tempo: 6,0081481933594E-5
    Execuções: 1
Pick Random: 39.9687%
    Tempo: 0,0050480365753174
    Execuções: 90
Rate: 1.7140%
    Tempo: 0,00021648406982422
    Execuções: 13
Remover Duplicados: 0.8891%
    Tempo: 0,00011229515075684
    Execuções: 3
Duplicar Plano de Corte (setNulos e removerNulosArr): 25.6031%
    Tempo: 0,0032336711883545
    Execuções: 20
  - Set Nulos: 8.1545%
      Tempo: 0,00026369094848633
      Execuções: 17
  - Remover Nulos Array: 42.4906%
      Tempo: 0,0013740062713623
      Execuções: 20
```

Figura 40 – Extrato parcial do log

5.1 Comparativos com outras soluções

Esta secção visa comparar a solução desenvolvida com outras identificadas no estado da arte. As aplicações são testadas com o mesmo conjunto de dados de entrada, para ser possível comparar os resultados e apurar conclusões. São utilizados os dados do perfil N12001, do anexo 2.

Os dados do problema são extraídos de uma relação de obra (ver anexo 2), e são os seguintes:

- Comprimento da barra: 650cm
- Material a cortar: 2x315, 4x200, 6x155, 2x150, 4x130, 2x105, 2x70, 3x75

Uma vez introduzidos os dados na aplicação desenvolvida, um dos diversos resultados encontrados, visto não existir o conceito de melhor solução, foi o que se pode observar na figura 41.

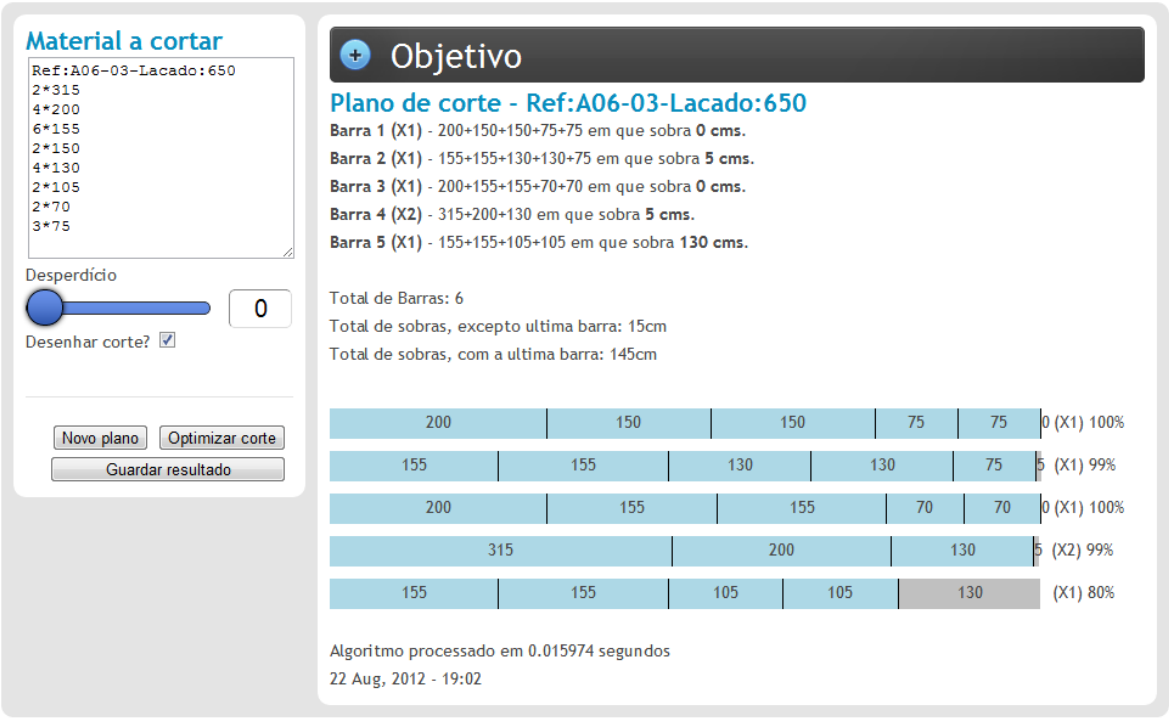


Figura 41 – simulação com a aplicação desenvolvida

Verifica-se um excelente aproveitamento de todas as barras utilizadas, com valores entre os 99% e 100%. A sobra da última barra tem comprimento suficiente para ser aproveitada para obras seguintes, não sendo assim considerada lixo ou desperdício.

Segue-se a simulação com a aplicação comercial “Corte Corte 1D”. Denote-se que cada aplicação tem as suas próprias especificidades quanto à introdução de dados. Desta forma,

também é possível observar a interface de cada uma das aplicações e comparar as diferenças que existem entre elas. A figura 42 demonstra a introdução dos mesmos dados na aplicação "Corte Certo 1D".

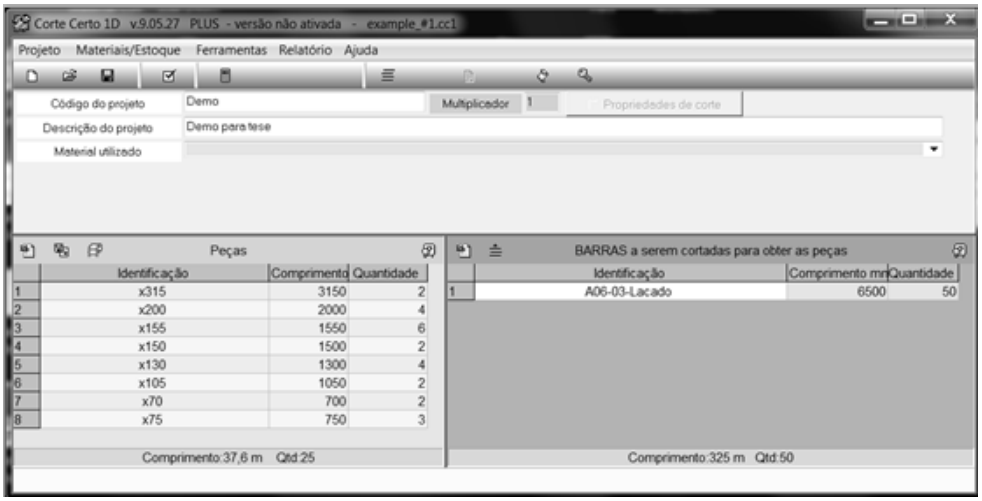


Figura 42 – Introdução de dados na aplicação Corte Certo 1D

O resultado obtido é o que se pode observar na figura 43. Pode-se desde logo verificar que o output produzido gera alguma confusão na interpretação da sequência de corte, tendo em conta a forma como esta é apresentada.

A informação apresentada pode facilmente levar a erros de perceção por parte do operador de corte, uma vez que facilmente se pode confundir a referência com a dimensão da peça a cortar. O tempo de execução foi bastante rápido, inferior a 1 segundo.

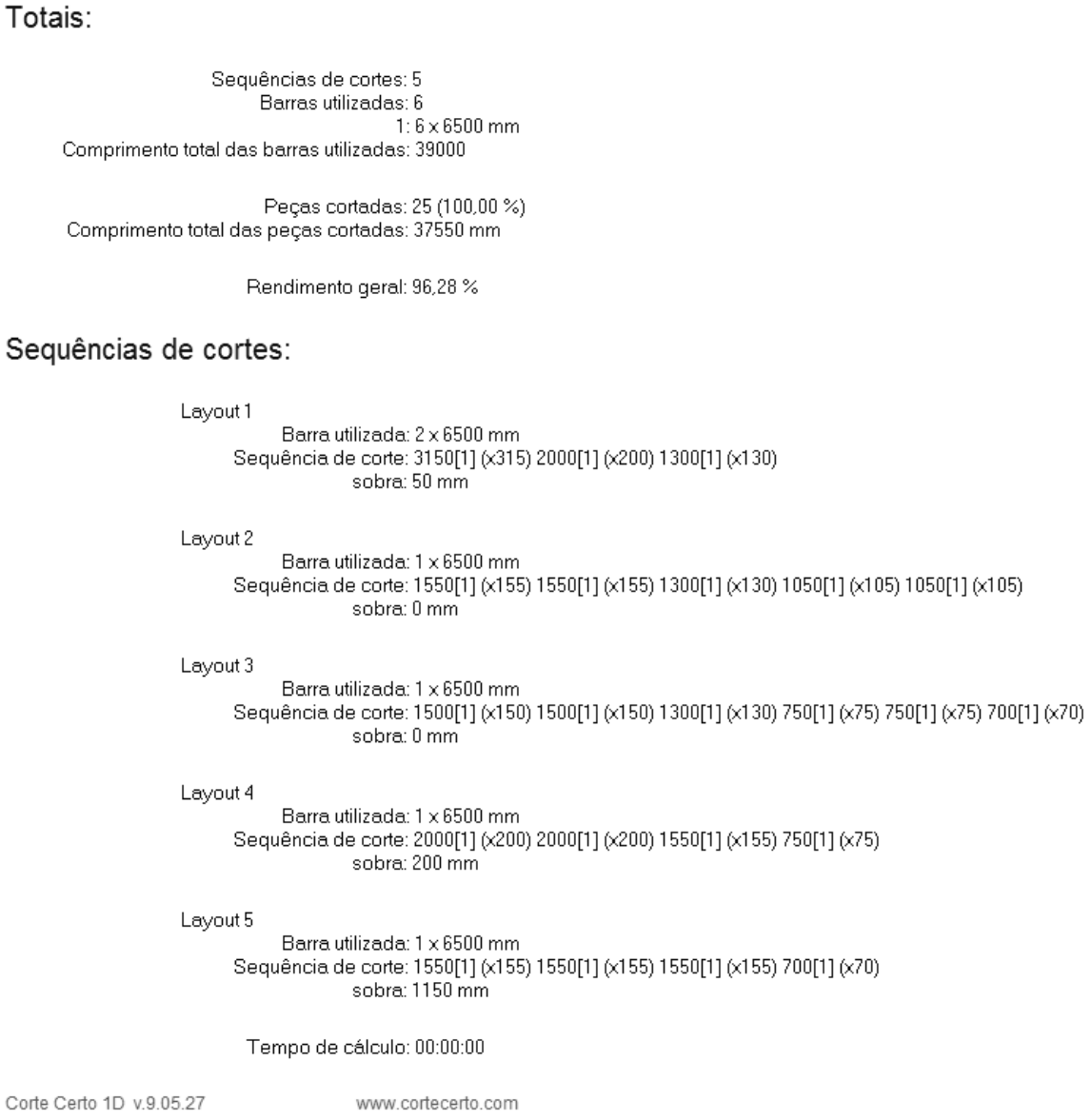


Figura 43 – simulação com a aplicação Corte Certo 1D

Outro software comercial testado foi o “1D Nesting Optimizer”. Os dados foram lançados na aplicação, conforme se observa na figura 44.

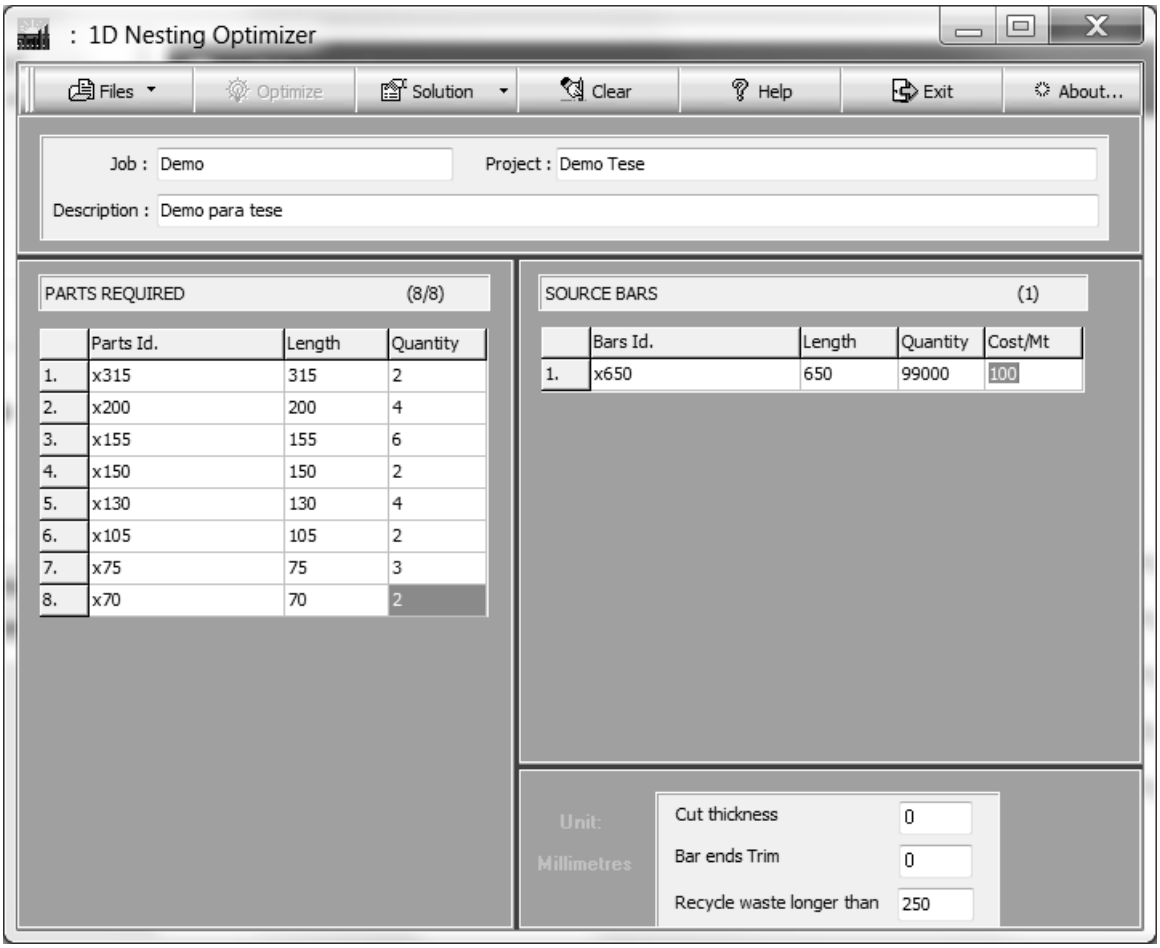
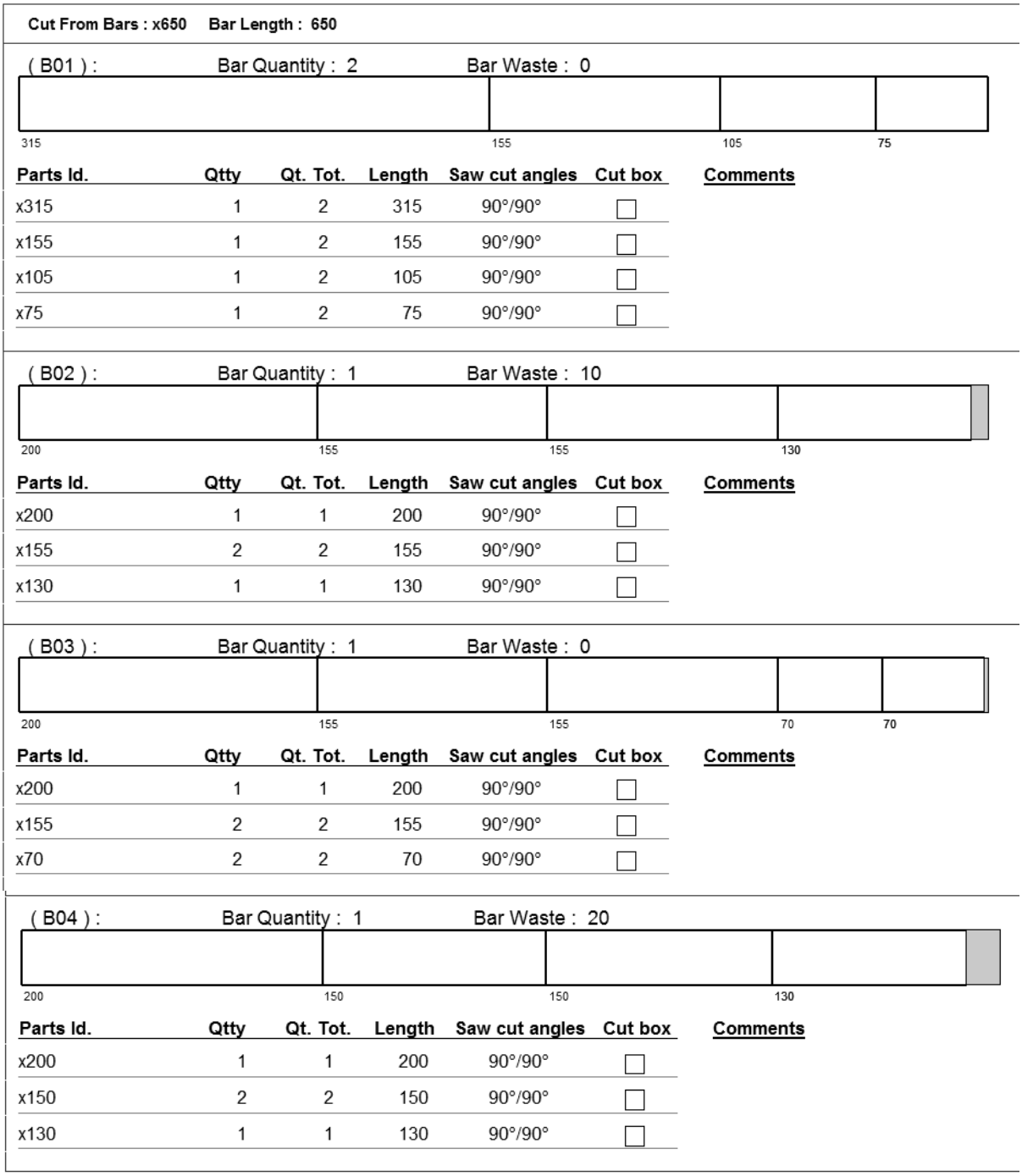


Figura 44 – Introdução de dados na aplicação 1D Nesting Optimizer

Também nesta aplicação o resultado é apresentado em menos de 1 segundo. De uma forma um pouco mais elaborada e detalhada que a aplicação anterior, o resultado é o apresentado nas figuras 45 e 46. Desde logo se pode observar que os resultados apresentados são um pouco extensos, face ao que é necessário facultar ao operador de corte.



Cut From Bars : x650 **Bar Length : 650**

(B01) : Bar Quantity : 2 Bar Waste : 0

315 155 105 75

Parts Id.	Qtty	Qt. Tot.	Length	Saw cut angles	Cut box	Comments
x315	1	2	315	90°/90°	☐	
x155	1	2	155	90°/90°	☐	
x105	1	2	105	90°/90°	☐	
x75	1	2	75	90°/90°	☐	

(B02) : Bar Quantity : 1 Bar Waste : 10

200 155 155 130

Parts Id.	Qtty	Qt. Tot.	Length	Saw cut angles	Cut box	Comments
x200	1	1	200	90°/90°	☐	
x155	2	2	155	90°/90°	☐	
x130	1	1	130	90°/90°	☐	

(B03) : Bar Quantity : 1 Bar Waste : 0

200 155 155 70 70

Parts Id.	Qtty	Qt. Tot.	Length	Saw cut angles	Cut box	Comments
x200	1	1	200	90°/90°	☐	
x155	2	2	155	90°/90°	☐	
x70	2	2	70	90°/90°	☐	

(B04) : Bar Quantity : 1 Bar Waste : 20

200 150 150 130

Parts Id.	Qtty	Qt. Tot.	Length	Saw cut angles	Cut box	Comments
x200	1	1	200	90°/90°	☐	
x150	2	2	150	90°/90°	☐	
x130	1	1	130	90°/90°	☐	

Figura 45 – Parte 1 dos resultados (1D Nesting Optimizer)

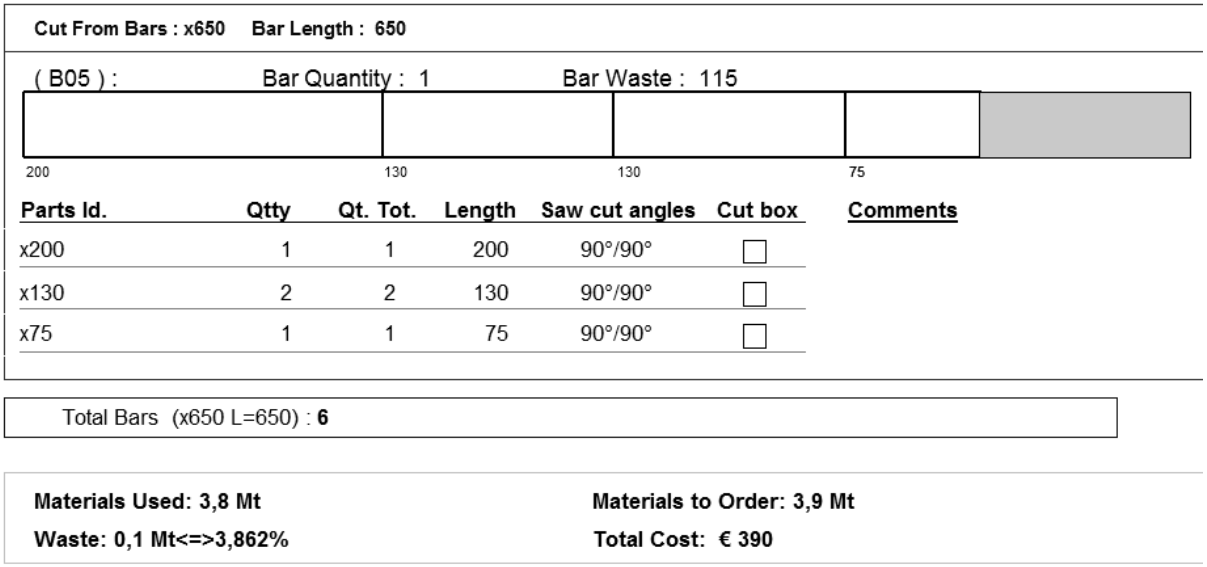


Figura 46 - Parte 2 dos resultados (1D Nesting Optimizer)

Para que seja possível ter uma correta perceção dos resultados obtidos até aqui, é necessário transpor os mesmos para um método equivalente de representação. Para tal, os resultados encontrados a partir das simulações executadas em cada uma das aplicações foram introduzidos no Excel, gerando os seguintes quadros de informação.

Quadro 6 – Resultados obtidos pela aplicação desenvolvida

Aplicação desenvolvida

Barra	Qt	% util	sobra	soma	Plano de corte					
Barra 1	1	100%	0	650	200	150	150	75	75	
Barra 2	1	99%	5	645	155	155	130	130	75	
Barra 3	1	100%	0	650	200	155	155	70	70	
Barra 4	2	99%	5	645	315	200	130			
Barra 5	1	80%	130	520	155	155	105	105		

Quadro 7 – Resultados obtidos pela aplicação Corte Certo 1D

Corte Certo 1D

Barra	Qt	% util	sobra	soma	Plano de corte					
Barra 1	2	99%	5	645	315	200	130			
Barra 2	1	100%	0	650	155	155	130	105	105	
Barra 3	1	100%	0	650	150	150	130	75	75	70
Barra 4	1	97%	20	630	200	200	155	75		
Barra 5	1	82%	115	535	155	155	155	70		

Quadro 8 – Resultados obtidos pela aplicação 1D Nesting Optimizer

1D Nesting Optimizer

Barra	Qt	% util	sobra	soma	Plano de corte					
Barra 1	2	100%	0	650	315	155	105	75		
Barra 2	1	98%	10	640	200	155	155	130		
Barra 3	1	100%	0	650	200	155	155	70	70	
Barra 4	1	97%	20	630	200	150	150	130		
Barra 5	1	82%	115	535	200	130	130	75		

A partir destes dados foi possível produzir os seguintes gráficos, que facilitam a leitura de resultados.

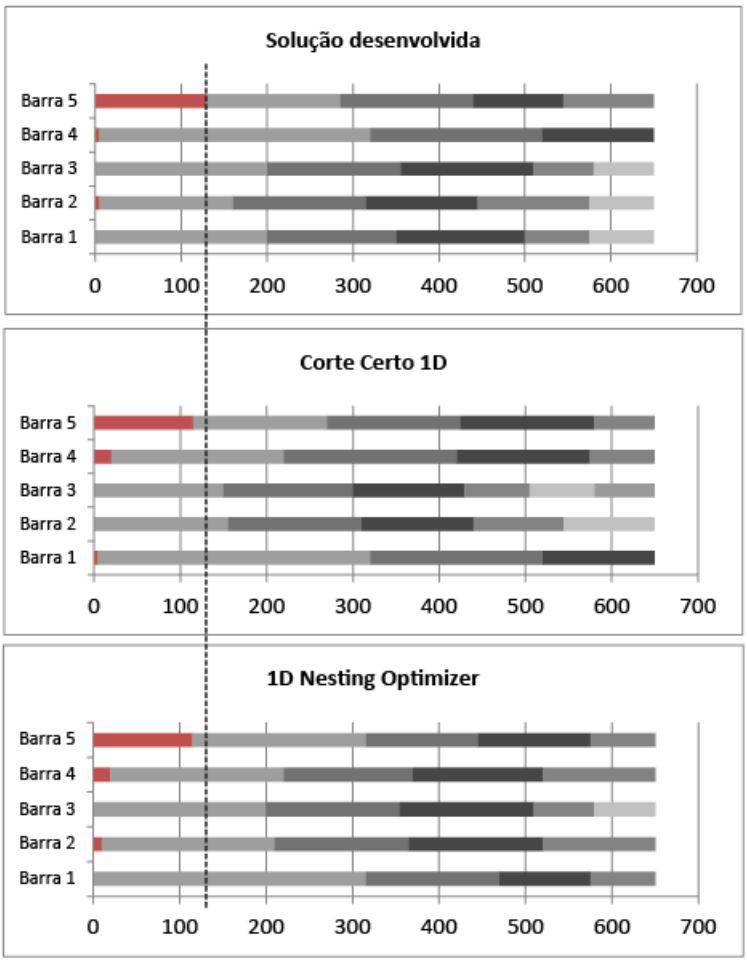


Figura 47 – Comparação entre resultados obtidos

A vermelho, encontram-se representadas as sobras em cada uma das barras. Ainda que a diferença não seja muito expressiva, a solução desenvolvida, quando parametrizada para minimizar o desperdício, consegue chegar a um resultado que pode ser considerado melhor, face às soluções comerciais. O aproveitamento de cada barra foi em média superior ao das

outras soluções, traduzindo-se num resto final, na última barra, que é 15cm mais comprido. À primeira vista, esta diferença pode não ser importante, mas hipoteticamente e se fosse necessário, seria possível cortar mais uma peça de 130cm sem qualquer desperdício. Se só sobrar uma peça de 115cm o melhor aproveitamento seria para cortar uma peça de 105cm onde se verifica um desperdício de 10cm.

São o somatório destas pequenas diferenças pontuais, que ao longo do tempo começam certamente a fazer alguma diferença em termos de otimização e rentabilização do material.

Para além dos factos apresentados no que toca à eficiência do plano de corte, graficamente a solução desenvolvida supera largamente as restantes, quer pela simplicidade de introdução dos dados, quer pelo output gerado para o ecrã ou impressão, com informação sobre os planos de corte a executar.

Por mera curiosidade e para validar os resultados obtidos, foi feita nova simulação em cada uma das aplicações. Apenas a solução desenvolvida gerou resultados distintos, e como qualquer solução obtida apenas é considerada uma boa solução e não a melhor, o resultado produzido desta vez é ainda melhor que o anterior.

Na figura 48 pode-se observar que para o mesmo conjunto de dados de entrada o algoritmo apresenta um resultado que utiliza o mesmo número de barras, mas com uma sobra na última barra de 140cm em vez de 130cm. Note-se que 4 barras têm uma taxa de aproveitamento de 100% e outra de 99%.

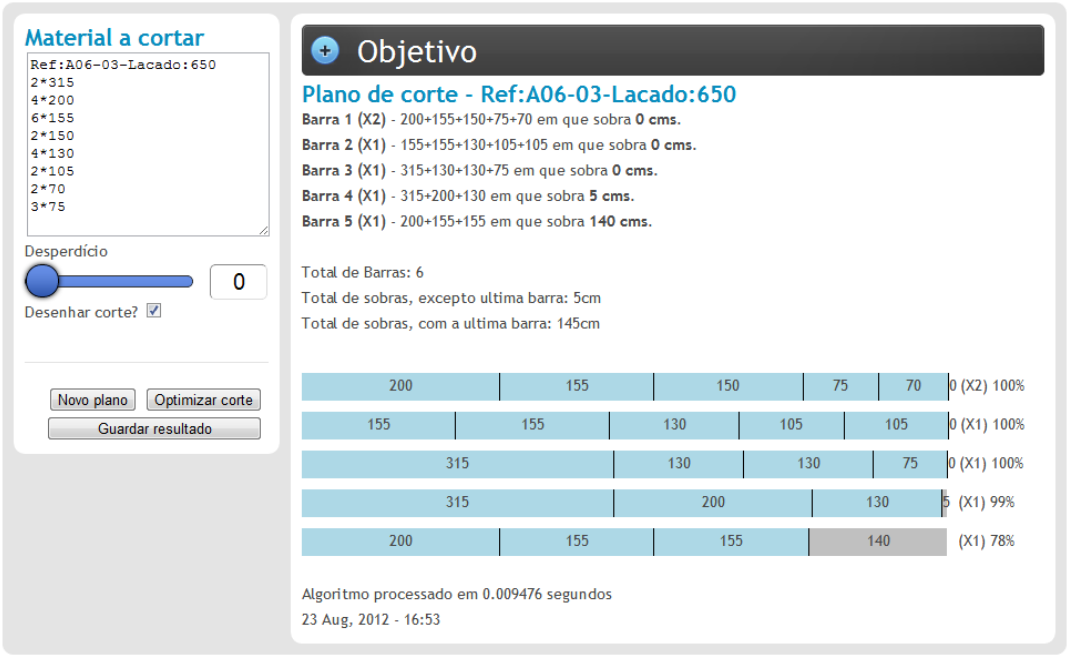


Figura 48 – Outro resultado obtido com a aplicação desenvolvida

Não foi possível executar comparações com as aplicações a seguir apresentadas, devido a restrições das versões de demonstração:

- Bar Cut Optimier
- Cut 1D x
- Plus 1D

De uma modo geral, as aplicações comerciais aqui analisadas, além de ser necessário indicar obviamente a lista de material a cortar, requerem também a introdução prévia de material fonte (stock), para ser possível proceder à execução da otimização. Também é pertinente referir que qualquer uma das soluções comerciais apresentadas, requer a sua instalação no computador onde se pretenda executar as simulações de otimização.

Quanto ao fator monetário, o quadro x apresenta uma tabela de preços, para as diversas soluções:

Quadro 9 – Comparativo de preços

Programa	**Fonte**	**Preço**
Corte Certo 1D[3]	www.cortecerto.com	>= 437,20 €
1D-Nest	www.1d-solutions.com	>= 153,00 €
Bar Cut Optimizer	www.binrace.com/bar-cut-optimizer	135,14 €
Cut 1D X	www.optimalprograms.com/cut_1d_x.htm	25,00 €
CutLogic 1D Trial	www.tmachines.com/index.htm	>= 237,68 €
Plus 1D[4]	www.nirvanatec.com/bar_nesting_software.html	249,00 €
Aplicação desenvolvida	www.jrgomes.net/corte1d	Gratuita

O anexo 3 contém novo comparativo entre aplicações com outro conjunto de dados provenientes da relação de obra presente no anexo 2.

[3] Disponível em português
[4] Disponível em português

6. Conclusões e trabalho futuro

Neste capítulo é encerrada esta dissertação recapitulando os objetivos que foram inicialmente propostos e possíveis aspetos a melhorar num trabalho posterior.

6.1 Objetivos alcançados

Esta dissertação assentou numa necessidade real, que pessoalmente se manifestou numa motivação acrescida. Presentemente a solução está a ser utilizada e a evidenciar as mais-valias e funcionalidades desenvolvidas.

É de todo útil e muito enriquecedor, sempre que possível, aplicar os conhecimentos adquiridos na resolução de problemas reais. Desta forma é possível contactar com as entidades, observar e sentir as necessidades, verificar o seu modo de funcionamento tradicional, recolher opiniões dos funcionários, etc..

A verdadeira satisfação advém de poder verificar os resultados em termos práticos e saber que o objetivo foi plenamente atingido, colmatando assim uma falha que existia num sector específico.

Recordando o início do problema, onde foram aplicadas várias horas e envolvidas diversas pessoas para se chegar a uma boa solução final (de forma manual), e verificar que agora numa curta fração de segundo temos disponível uma das melhores soluções possíveis, por si só já se traduz num enorme sucesso. Mais aliciante é o facto da solução desenvolvida, conseguir

produzir melhores resultados que soluções profissionais já existentes no mercado há alguns anos.

Para o problema exposto no início desta dissertação, eis uma possível solução, calculada em menos de uma décima de segundo, que comparativamente à solução manual, apresenta menor desperdício:

Plano de corte - Ref:A06-03-Lacado:650

Barra 1 (X3) - 250+180+130+90 em que sobra **0 cms.**
Barra 2 (X1) - 130+130+100+100+100+90 em que sobra **0 cms.**
Barra 3 (X1) - 130+130+100+100+90+90 em que sobra **10 cms.**
Barra 4 (X1) - 180+130 em que sobra **340 cms.**

Método manual apresentado pelo funcionário com bastantes anos de experiência:

- 1 Perfil de 6,50 -> (2 x 2,50) + (1 x 1,30) sobra 0,20 de desperdício
- 1 Perfil de 6,50 -> (3 x 1,80) + (1 x 1,00) sobra 0,10 de desperdício
- 1 Perfil de 6,50 -> (6 x 0,90) + (1 x 1,00) sobra 0,10 de desperdício
- Etc…

6.2 Melhorias futuras

Neste momento a solução recebe a referência de um determinado perfil de material e o seu comprimento total. Isto faz com que sejam usadas sempre barras com o tamanho definido para esse perfil.

Seria interessante, permitir introduzir uma lista de material disponível com os desperdícios de outras obras, para se começar a gastar primeiro os restos de material e só em último caso usar barras novas. Esta situação é contornável na versão atual, inserindo diversas vezes o mesmo perfil com tamanhos distintos.

A curto prazo está previsto otimizar a interface da solução para dispositivos móveis.

Numa fase posterior, está previsto demonstrar e comprovar estatisticamente a mais-valia da solução desenvolvida. Para isso pretende-se analisar os últimos trabalhos realizados por uma empresa em que o processo de otimização apenas se baseia na experiência do serralheiro e é realizado sem a ajuda de qualquer meio computacional. Os mesmos dados comparam a

solução obtida com a solução gerada pela aplicação desenvolvida no sentido de se avaliarem as diferenças e eventuais ganhos que podem advir da sua utilização.

6.3 Considerações finais

Ao longo deste documento foi possível constatar a evolução e maturação do projeto até esta sua fase final. Foram diversos os desafios até conseguir a melhor orientação para a solução implementada. Todos os esforços enveredados neste projeto, todas as fases, todas as tentativas, algumas falhadas, as correções, os contributos de serralheiros, orientadores, professores, permitiram ter sucesso no resultado conseguido. Tudo aquilo que pode ser dito nesta fase é um pouco repetitivo daquilo que já foi dito ao longo deste documento, no que diz respeito à qualidade do produto obtido.

À semelhança da comparação feita no capítulo 2 com as provas de atletismo ao referenciar o conceito de limite superior, é caso para dizer que também esta solução diminui este valor, batendo um novo recorde, não só pelos resultados que se conseguem obter, quando se privilegia o minimizar do desperdício, mas também pelo facto do utilizador poder ter controlo e decidir aquele que é para si o melhor plano de corte. Tudo isto, aliado à simplicidade da interface e à sua disponibilização na web, tornando-a acessível a partir de qualquer dispositivo com conectividade à internet, são factos que deixam fortes expectativas para uma utilização mais massiva e intensa por parte dos utilizadores a quem se destina.

REFERÊNCIAS

[Vanderbeck, F. 2000] Exact algorithm for minimizing the number of setups in the one-dimensional cutting stock problem.

[Valério de Carvalho 1999] Exact solution of bin-packing problems using column generation branch-and-bound

[Garey e Johnson 1979] Garey, M. and Johnson, D. 1979. Computers and Intractability. A Guide to the Theory of NP-Completeness. Freemann, San Francisco.

[Dyckhoff, H. 1990] A typology of cutting and packing problems. European Journal of Operational Research.

[Johnson e McGeoch 1997] Johnson, D. S. and L. A. McGeoch, 1997. The Traveling Salesman Problem: A Case Study. In E. H. L. Aarts and J. K. Lenstra, eds., Local Search in Combinatorial Optimization, 215-310, Wiley & Sons, New York.

[Holland 1975] Holland, J. H., 1975. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, University of Michigan Press (second edition: MIT Press, 1992).

[Goldberg 1989] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

Morgado, Francisco, 2010-11, *Acetatos de apoio à disciplina de Tópicos Avançados em Algoritmos*, Escola Superior de Tecnologia de Viseu

Loureiro**,** Jorge, 2010-11, *Acetatos de apoio à disciplina de Computação Inspirada em Vida*, Escola Superior de Tecnologia de Viseu

Bronson, Richard, Naadimuthu, Govindasami, 2001, *Investigação Operacional 2ª edição*, McGraw-Hill (Tradução portuguesa por Ruy Costa)

Ziviani, Nivio, *Projecto de Algoritmos com Implementações em Java e C++* [online]. Consultado em Junho 2012. Disponível na www. URL: http://www.dcc.ufmg.br/algoritmos-java/cap9/transp/completo1/cap9.pdf

Algoritmos genéticos, Introdução aos Algoritmos Genéticos [online]. [consultado em Abril de 2012]. Disponível na World wide Web: <http://professor.webizu.org/ga/>

Algoritmos genéticos, Definição [online]. [consultado em Abril de 2012]. Disponível na World wide Web: <http://pt.wikipedia.org/wiki/algoritmos_genéticos>

Fundamentos dos algoritmos genéticos [online]. [consultado em Maio de 2012]. Disponível na World wide Web: <http://www.docstoc.com/docs/55403374/>

ANEXO 1

Exemplo do output produzido pelas aplicações utilizadas na Martifer Alumínios.

MARTIFER ALUMÍNIOS [redacted]	Job/Group:	1058037B
	Client/Site:	/
	Description:	Caixilhos Soleal
	Printout date/time:	[redacted]
	Page:	29

JOB BARCODE !a05330|PPPPPp

OPTIMIZATION REPORT

215031 (FYSOLEAL) 1-Fabricante mate ET - R7043 R7043 - Cinzento	19 bars, length 6500	Total length: 123.5 m - Total weight: 0.0 kg Usage = 85.5% ReCoverable waste = 10.8% Scrap = 3.7%
215078 (FYSOLEAL) 1-Fabricante mate ET - R7043 R7043 - Cinzento	7 bars, length 6500	Total length: 45.5 m - Total weight: 0.0 kg Usage = 88.7% ReCoverable waste = 9.2% Scrap = 2.1%
215194 (FYSOLEAL) 1-Fabricante mate ET - R7043 R7043 - Cinzento	17 bars, length 6500	Total length: 110.5 m - Total weight: 0.0 kg Usage = 94.2% ReCoverable waste = 1.9% Scrap = 3.9%
215204 (FYSOLEAL) 1-Fabricante mate ET - R7043 R7043 - Cinzento	3 bars, length 6500	Total length: 19.5 m - Total weight: 0.0 kg Usage = 83.0% ReCoverable waste = 11.3% Scrap = 5.7%
215252 (FYSOLEAL) 1-Fabricante mate ET - R7043 R7043 - Cinzento	2 bars, length 6500	Total length: 13.0 m - Total weight: 0.0 kg Usage = 72.7% ReCoverable waste = 26.7% Scrap = 0.6%
591003 (FYSOLEAL) 1-Fabricante mate ET - R7043 R7043 - Cinzento	14 bars, length 6500	Total length: 91.0 m - Total weight: 0.0 kg Usage = 92.7% ReCoverable waste = 4.3% Scrap = 3.0%
881000 (FYSOLEAL)	17 bars, length 6500	Total length: 110.5 m - Total weight: 0.0 kg Usage = 92.9% ReCoverable waste = 4.7% Scrap = 2.5%

Total weight: 0.0 kg

MARTIFER ALUMINIO	Job/Group:	1058037B
	Client/Site:	/
	Description:	Caixilhos Soleal
	Printout date/time:	[redacted]
	Page:	30

JOB BARCODE !a05330|PPPPPp

OPTIMIZED CUTTING LIST - Cutting cicles 379

PROFILES

215078 (FYSOLEAL) 1-Fabricante mate ET - R7043 R7043 - Cinzento — Total length: 45.5 m - Total weight: 0.0 kg

1 bar, length 6500	6060 - 01.0533.03.0002.FRA.W	x 1	135 919° [919] 135
	6060 - 01.0533.03.0004.FRA.W	x 1	135 919° [919] 135
	6024 - 01.0533.01.0002.FRA.W	x 1	135 750° [750] 135
	6024 - 01.0533.01.0004.FRA.W	x 5	135 750° [750] 135 — Residue = 74.8 (lost)
2 bars, length 6500	6024 - 01.0533.01.0001.FRA.H	x 3	135 2110° [2110] 135 — Residue = 118.5 (lost)
1 bar, length 6500	6024 - 01.0533.01.0001.FRA.H	x 2	135 2110° [2110] 135
	6060 - 01.0533.03.0001.FRA.H	x 1	135 2110° [2110] 135 — Residue = 118.5 (lost)
1 bar, length 6500	6060 - 01.0533.03.0001.FRA.H	x 1	135 2110° [2110] 135
	6024_2 - 01.0533.02.0001.FRA.H	x 2	135 2110° [2110] 135 — Residue = 118.5 (lost)
1 bar, length 6500	6024 - 01.0533.01.0001.FRA.H	x 1	135 2110° [2110] 135
	6024 - 01.0533.01.0002.FRA.W	x 4	135 750° [750] 135
	6024_2 - 01.0533.02.0002.FRA.W	x 1	135 745° [745] 135 — Residue = 572.1 (reusable)
1 bar, length 6500	6024 - 01.0533.01.0001.FRA.H	x 1	135 2110° [2110] 135
	6024_2 - 01.0533.02.0004.FRA.W	x 1	135 745° [745] 135 — Residue = 3600.7 (reusable)

215194 (FYSOLEAL) 1-Fabricante mate ET - R7043 R7043 - Cinzento — Total length: 110.5 m - Total weight: 0.0 kg

1 bar, length 6500	6024 - 01.0533.01.0015.COP.H	x 1	135 2026° [2026] 135
	5910 - 01.0533.06.0021.COP.H	x 1	135 1343° [1343] 135
	5918 - 01.0533.07.0021.COP.H	x 1	135 1343° [1343] 135
	5910 - 01.0533.06.0019.COP.W	x 1	135 856° [856] 135
	6060 - 01.0533.03.0017.COP.W	x 1	135 835° [835] 135 — Residue = 31.2 (lost)
1 bar, length 6500	6024 - 01.0533.01.0015.COP.H	x 1	135 2026° [2026] 135
	7096 - 01.0533.09.0033.COP.H	x 2	135 1343° [1343] 135

(*) Low Corner, Wall Nib (Between square brackets: cutting quote with no counter block)

Date: [redacted]
Group/Job: **1058026C**
Description:

Serie:	**MX**
Code:	**JM150A**
Description:	
Ext. Colour:	**600**
Int. Colour:	**600**
Profile utilization:	**96,0%**
Scrap as offcuts:	**3,7%**
Scrap not reusable:	**0,3%**

#	Bars	Mul.	Length	Surplu	Nr.	Length.	Ident.	LHH	RHH	LHB	RHB	Pos.Qu
1	1	1	6500,0	11,2	1	2486,4	03.0066.02.MNT.H	90,0	90,0	90,0	90,0	2486,4
					1	3981,4	04.0024.02.MNT.H	90,0	90,0	90,0	90,0	3981,4
2	1	1	6500,0	11,2	1	2486,4	03.0065.02.MNT.H	90,0	90,0	90,0	90,0	2486,4
					1	3981,4	04.0013.02.MNT.H	90,0	90,0	90,0	90,0	3981,4
3	1	1	6500,0	11,2	1	2486,4	03.0080.02.MNT.H	90,0	90,0	90,0	90,0	2486,4
					1	3981,4	04.0055.02.MNT.H	90,0	90,0	90,0	90,0	3981,4
4	1	1	6500,0	11,2	1	2486,4	03.0072.02.MNT.H	90,0	90,0	90,0	90,0	2486,4
					1	3981,4	04.0050.02.MNT.H	90,0	90,0	90,0	90,0	3981,4
5	1	1	6500,0	11,2	1	2486,4	03.0002.02.MNT.H	90,0	90,0	90,0	90,0	2486,4
					1	3981,4	04.0062.02.MNT.H	90,0	90,0	90,0	90,0	3981,4
6	1	1	6500,0	11,2	1	2486,4	03.0034.02.MNT.H	90,0	90,0	90,0	90,0	2486,4
					1	3981,4	04.0018.02.MNT.H	90,0	90,0	90,0	90,0	3981,4
7	1	1	6500,0	15,6	1	211,0	11.0165.02.TRV.L	90,0	90,0	90,0	90,0	211,0
					1	218,0	08.0163.02.TRV.L	90,0	90,0	90,0	90,0	218,0
					1	1760,0	08.0208.02.TRV.L	90,0	90,0	90,0	90,0	1760,0
					1	4263,4	11.0056.02.MNT.H	90,0	90,0	90,0	90,0	4263,4
8	1	1	6500,0	22,6	1	211,0	11.0139.02.TRV.L	90,0	90,0	90,0	90,0	211,0
					1	211,0	11.0158.02.TRV.L	90,0	90,0	90,0	90,0	211,0
					1	1760,0	08.0177.02.TRV.L	90,0	90,0	90,0	90,0	1760,0
					1	4263,4	11.0057.02.MNT.H	90,0	90,0	90,0	90,0	4263,4
9	1	1	6500,0	23,5	1	557,0	08.0215.02.TRV.L	90,0	90,0	90,0	90,0	557,0
					1	557,0	08.0118.02.TRV.L	90,0	90,0	90,0	90,0	557,0
					1	557,0	08.0174.02.TRV.L	90,0	90,0	90,0	90,0	557,0
					1	758,0	09.0087.02.TRV.L	90,0	90,0	90,0	90,0	758,0
					1	4010,0	04.0009.02.MNT.H	90,0	90,0	90,0	90,0	4010,0
10	1	1	6500,0	24,5	1	607,0	01.0085.02.TRV.L	90,0	90,0	90,0	90,0	607,0
					1	607,0	01.0188.02.TRV.L	90,0	90,0	90,0	90,0	607,0
					1	607,0	01.0100.02.TRV.L	90,0	90,0	90,0	90,0	607,0
					1	607,0	01.0202.02.TRV.L	90,0	90,0	90,0	90,0	607,0
					1	4010,0	04.0044.02.MNT.H	90,0	90,0	90,0	90,0	4010,0
11	1	1	6500,0	44,0	1	166,0	08.0223.02.TRV.L	90,0	90,0	90,0	90,0	166,0
					1	218,0	08.0164.02.TRV.L	90,0	90,0	90,0	90,0	218,0
					1	1760,0	09.0209.02.TRV.L	90,0	90,0	90,0	90,0	1760,0
					1	4280,0	11.0039.02.MNT.H	90,0	90,0	90,0	90,0	4280,0

ANEXO 2

Relação de uma obra da empresa Gilberto Jesus Figueiredo, utilizada para lançar dados concretos na aplicação.

GILBERTO JESUS FIGUEIREDO
COVA DA SERPE - QUIAIOS
3080 FIGUEIRA DA FOZ
Telef. 033 910294
910119

Data ..2-12-2011

Sr. [redacted]
Figueira da Foz

Alumínio Lacado 9007 Mate Navarra 12000

Listagem de peças a cortar:

Perfil N12001

4	130
4	200
2	35
2	105
2	150
2	155
2	70
2	155
2	315
2	155

Perfil N12025

8	200
8	65
4	150
4	77
2	70
10	155
8	78

Perfil N12076

2	200
1	150

Perfil N12052

2	30
2	105

Perfil N12011

3	155

ANEXO 3

Demonstração de resultados utilizando o conjunto de dados para o perfil N12025 do anexo 2.

Aplicação desenvolvida

Barra	Qt	% util	sobra	soma	Plano de corte							
Barra 1	4	100%	2	648	200	155	150	78	65			
Barra 2	1	100%	0	650	200	155	155	70	70			
Barra 3	2	99%	9	641	200	155	78	78	65	65		
Barra 4	1	95%	32	618	155	155	77	77	77	77		
Barra 5	1	31%	450	200	200							

Total barras	9			
Média s/ ultima barra		98%	43	Total sobra s/ ultima barra
Média total		85%	493	Total sobra

CutLogic 1-D

Barra	Qt	% util	sobra	soma	Plano de corte							
Barra 1	4	100%	3	647	200	155	150	77	65			
Barra 2	2	99%	4	646	200	155	78	78	70	65		
Barra 3	1	99%	9	641	200	155	78	78	65	65		
Barra 4	1	96%	29	621	155	155	155	78	78			
Barra 5	1	31%	450	200	200							

Total barras	9			
Média s/ ultima barra		98%	45	Total sobra s/ ultima barra
Média total		85%	495	Total sobra

Corte Certo 1D

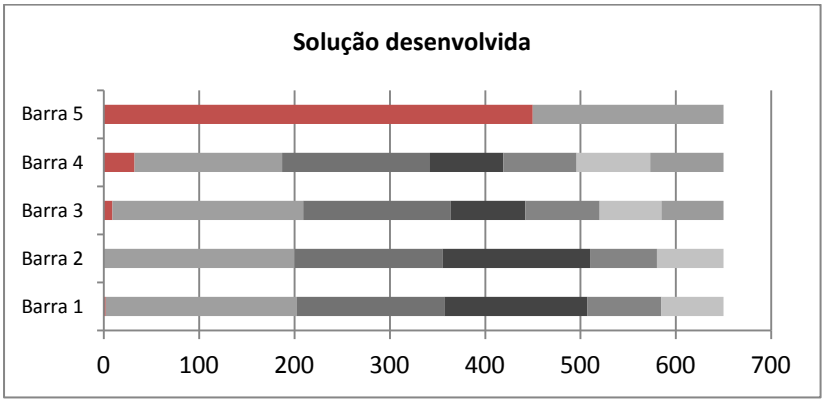
Barra	Qt	% util	sobra	soma	Plano de corte							
Barra 1	1	100%	0	650	200	150	150	150				
Barra 2	3	97%	17	633	200	200	155	78				
Barra 3	1	100%	0	650	200	155	155	70	70			
Barra 4	1	99%	6	644	150	78	78	78	65	65	65	65
Barra 5	1	100%	2	648	155	155	78	65	65	65	65	
Barra 6	1	95%	30	620	155	155	155	78	77			
Barra 7	1	36%	419	231	77	77	77					

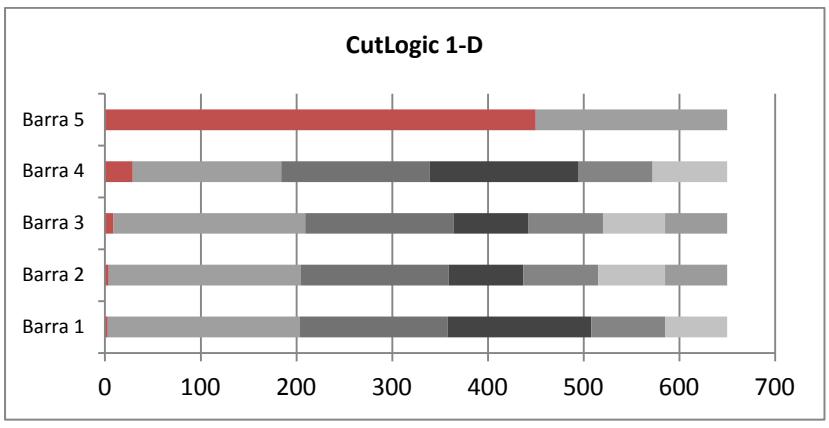
Total barras	9			
Média s/ ultima barra		99%	55	Total sobra s/ ultima barra
Média total		90%	474	Total sobra

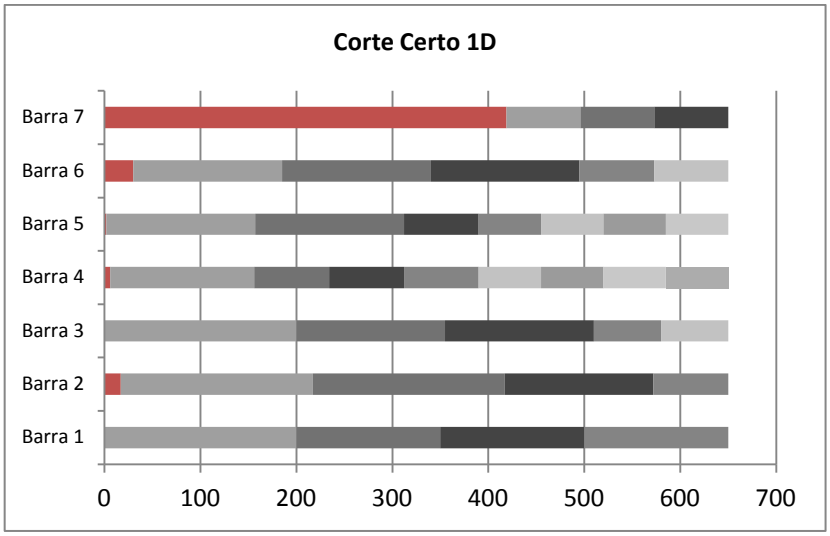
1D Nesting Optimizer

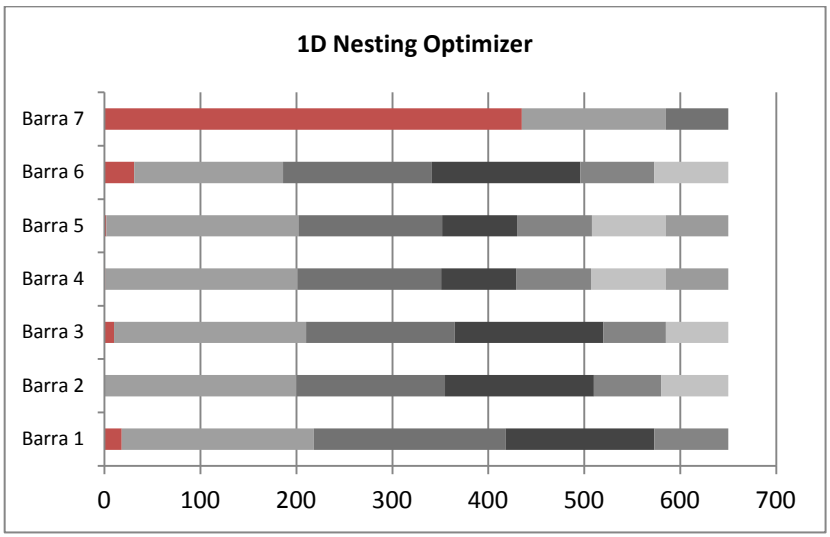
Barra	Qt	% util	sobra	soma	Plano de corte							
Barra 1	1	97%	18	632	200	200	155	77				
Barra 2	1	100%	0	650	200	155	155	70	70			
Barra 3	2	98%	10	640	200	155	155	65	65			
Barra 4	2	100%	1	649	200	150	78	78	78	65		
Barra 5	1	100%	2	648	200	150	78	78	77	65		
Barra 6	1	95%	31	619	155	155	155	77	77			
Barra 7	1	33%	435	215	150	65						
Total barras	9											
Média s/ ultima barra		98%	62	Total sobra s/ ultima barra								
Média total		89%	497	Total sobra								

Análise gráfica.









Conclusões:

- De um modo geral, todas as aplicações produzem boas soluções;
- Analisando as médias (com e sem a ultima barra), os resultados são muito equivalentes;
- Sem qualquer configuração adicional, apenas a aplicação desenvolvida e a *CutLogic 1D*, obtiveram uma sobra na última barra de 450cm face aos 419cm e 435cm obtidos pelas aplicações *Corte Certo 1D* e *1D Nesting Optimizer*, respetivamente;
- Todas as soluções utilizaram o mesmo número de barras;
- As duas primeiras aplicações utilizam 5 planos distintos de corte, face aos 7 gerados pelas outras duas.

- Tendo em conta o enunciado do problema apresentado na introdução do capítulo 1, onde apenas sobras com tamanho igual ou superior a 80cm são consideradas uteis para obras seguintes, a solução desenvolvida é aquela que consegue desperdiçar menos material, com uma diferença de 2cm para a aplicação *CutLogic 1-D*, que lhe sucede, já que ambas conseguem garantir a maior sobra na ultima barra, com 450cm.