ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO

TRABALHO DE FORMATURA

PROGRAMAÇÃO DA PRODUÇÃO DE UMA LAMINAÇÃO UTILIZANDO SIMULATED ANNEALING

aluno: PAULO HENRIQUE BARROS SILVA

orientador: MIGUEL CEZAR SANTORO

1996

Aos meus pais

AGRADECIMENTOS

Ao humanitário professor Miguel Cezar Santoro, pela bem humorada e dedicada orientação.

À professora Celma de Oliveira Ribeiro, pela orientação, apoio e incentivo.

Aos engenheiros Furtado e Garrone, pelas informações fornecidas.

Aos amigos Carlos Benittes e Mário Lacroix, pela eficiente ajuda com a misteriosa linguagem de programação C.

Aos amigos Marcelo Lauretto e Cláudio, pelos inúmeros 'galhos quebrados'.

À Tami Maekawa, pela amizade, notas de aula e pelo serviço de encadernação personalizado.

Aos companheiros de moradia, em especial ao 207, pela impagável ajuda e pela febril convivência.

Aos amigos Leo e Matheus, pela amizade, compreensão e paciência.

À minha Jeanne, por tudo.

A todos que de alguma forma contribuíram para a realização deste trabalho.

SUMÁRIO

Neste trabalho foi realizado um estudo sobre a utilização do método de busca Simulated Annealing na programação da produção de um processo de fundição seguido de laminação.

Para tanto foi desenvolvido um programa de computador em linguagem C, que, utilizando tal método, busca a solução ótima através da geração e avaliação sucessiva de alternativas para a sequenciação das ordens de produção nas duas principais etapas do processo.

Uma carteira de pedidos e dados sobre os tempos de produção da empresa na qual o trabalho foi baseado foram obtidos para a utilização do programa. Pode-se, assim, executá-lo e avaliar os resultados fornecidos.

ÍNDICE

Capítulo 1. CARACTERIZAÇÃO DO PROBLEMA
1.1 Introdução 2
1.2 Descrição breve do processo 2
1.3 O pedido de compra 3
1.4 Características do processo de maior relevância 4
1.5 O problema da programação da produção 5

Capítulo 2. FUNDAMENTOS TEÓRICOS
2.1 Introdução 8
2.2 Descrição do Simulated Annealing 8
2.3 Analogia física 9
2.4 Conclusões teóricas 10
2.5 Implementação do Simulated Annealing 11
2.5.1 Escolhas específicas ao problema 11
2.5.2 Escolhas genéricas 12
2.6 Objetivo do trabalho 13

Capítulo 3. MODELAGEM DO PROBLEMA
3.1 Introdução 16
3.2 Considerações gerais 16
3.3 Sequenciações consideradas 16
3.4 Definição da solução 17
3.5 Solução para a fundição 17
3.6 Solução para laminação 18
3.7 Tempos de produção 21
3.8 Avaliação de desempenho 21

Capítulo 4. O SIMULATED ANNEALING
4.1 Introdução 26
4.2 Geração de solução vizinha 26
4.2.1 Na fundição 28
4.2.2 Na laminação 29
4.3 Geração de solução semente 29
4.3.1 Para a fundição 29
4.3.2 Para a laminação 30
4.4 Tabela de resfriamento 30
4.4.1 Para o Simulated Annealing completo 31
4.4.2 Para o Simulated Annealing da laminação 32

Capítulo 5. IMPLEMENTAÇÃO DO PROGRAMA
5.1 Introdução 35
5.2 Equipamento e programas utilizados 35
5.3 Entrada dos dados 36
5.4 Rotinas desenvolvidas 37
5.5 Testes realizados 46

Capítulo 6. SIMULAÇÃO
6.1 Introdução 49
6.2 Dados utilizados 49
6.2.1 Ordens de produção 49
6.2.2 Tempos de produção 49
6.2.3 Outros 50
6.3 Resultados obtidos 51
6.3.1 Primeira execução 51
6.3.2 Segunda execução 54
6.2.4 Teceira execução 58

Capítulo 7. CONCLUSÕES
7.1 Conclusões e proposta para continuidade 60

BIBLIOGRAFIA 63

Anexo 1: Código fonte A2
Anexo 2: Dados utilizados A24
Anexo 3: Resultados A36

1.

CARACTERIZAÇÃO DO PROBLEMA

1.1 INTRODUÇÃO

Este trabalho foi baseado no processo de produção de perfis de aço laminados de uma grande industria de base do estado de São Paulo, mais especificamente, de aço rápido, aço ferramenta, aço inoxidável, aço válvula e algumas outras ligas especiais, fabricados numa unidade situada no interior do estado. Doravante, a empresa será chamada de Lamina SA.

Seu objetivo não foi modelar fiel e rigorosamente este processo produtivo. O mesmo tem interligações com o processo produtivo de outros produtos, formando um conjunto de roteiros de produção bastante complexo e que deveria ser considerado como um todo numa simulação mais precisa, tornando o problema bastante extenso.

Em verdade, o real objetivo do trabalho foi avaliar de perto o comportamento do método de busca Simulated Annealing aplicado a um problema de programação da produção com fortes características de um problema real. Para tanto, foram feitas simplificações no roteiro e estimativas, com colaboração do pessoal da Lamina, de vários dados para execução do modelo.

No intuito de apresentar as característica do problema, esse capítulo descreve o processo de interesse, isto é, o de produção dos perfis citados, apresenta a forma de venda dos produtos da empresa, pormenoriza os aspectos do processo de maior relevância para a modelagem e discute as dificuldades em se realizar a programação da produção neste caso.

1.2 DESCRIÇÃO BREVE DO PROCESSO

A produção dos perfis laminados citados tem como primeira etapa a obtenção da liga metálica desejada. Isto é feito através da fundição em forno convencional de sucata e elementos de liga puros, tais como níquel, estanho, zinco, etc.

Inicialmente, a sucata é separada em baias de acordo com sua composição. Através de programação linear é determinada a quantidade de sucata de cada baia e a quantidade de elementos de liga puros, os quais possuem custo muito mais elevado, a serem utilizados para se chegar à composição da liga desejada. Essa combinação é então fundida. Em certo ponto da fundição, é retirada uma amostra para análise da

composição. Caso ainda não seja a desejada, mais elementos de liga puros são novamente adicionados.

Após a fundição, o metal é novamente solidificado em forma de lingotes. A próxima etapa do processo é o desbaste dos lingotes. Trata-se de uma primeira laminação, realizada num laminador dedicado a esta etapa.

Em seguida, é realizada a laminação propriamente dita. Os tarugos desbastados são passados no laminador para obtenção do perfil desejado para o produto final: barra chata, quadrada, sextavada, cilíndrica, etc.

A última fase do processo é a de acabamento. Nela os roteiros são bastante variados em função das diferentes necessidades dos clientes. Pode haver operações de corte, trefilação, torneamento, retífica e outras.

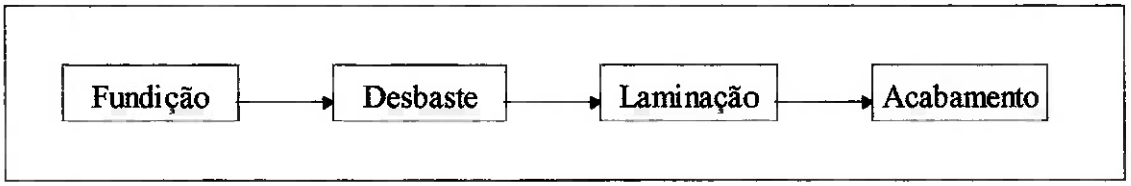

Figura 1.1: Roteiro de processo para perfis de aço rápido, ferramenta, inox e outros.

1.3 O PEDIDO DE COMPRA

A Lamina recebe pedidos de seus clientes contendo um ou mais itens. Cada item especifica:

A **liga metálica** desejada. Na empresa é identificado o grupo de ligas a que esta liga pertence. Assim, a informação sobre a liga fica sendo composta por grupo e liga propriamente dita. Por exemplo: aço ferramenta grupo 13 liga VD2 ou aço inoxidável grupo 21 V316. Ao todo foram consideradas aproximadamente 30 ligas, pertencentes a 5 grupos diferentes. Existem outras, além dessas, mas seguem roteiros diferentes daquele apresentado no item anterior.

As **dimensões do material** desejadas, apontando o tipo de perfil (barra chata, sextavada, etc.), as dimensões deste e o comprimento. Da mesma forma que na liga metálica, as dimensões do material também são atribuídas a grupos de dimensões, chamados faixas de bitola, que são importantes na etapa de laminação dos tarugos.

O **acabamento** desejado, podendo ser bruto, retificado, trefilado, torneado, fresado ou descascado.

A **quantidade desejada**, sendo, em média de cerca de 1.000 kg.

A **data de entrega**, combinada após negociação.

A importância do cumprimento de tal data varia em função do cliente e do tamanho do pedido. Pode-se dar maior prioridade a clientes novos e promissores em detrimento daqueles que sempre compram poucas quantidades ou até mesmo, caso necessário, daqueles com os quais a empresa já tenha um relacionamento menos abalável.

Por outro lado, o término de um pedido em data anterior à solicitada pelo cliente também é prejudicial para a empresa, já que, na maioria dos casos, não é possível realizar a entrega e o faturamento, ocasionando perdas financeiras e necessidade armazenamento do produto acabado.

Cada item desses pedidos gera uma ordem de produção, à qual está associada, da mesma forma que aos itens dos pedidos, o grupo e a liga, as dimensões, o acabamento, a quantidade e a data de entrega do produto.

1.4 CARACTERÍSTICAS DO PROCESSO DE MAIOR RELEVÂNCIA

Conforme havia ficado implícito no item 1.2, o forno processa lotes de mesma liga, tendo uma capacidade máxima de 25 t. Além disso, existe um limite mínimo de 10 t de ocupação para seu funcionamento, devendo ser obedecido para evitar a danificação das suas paredes internas.

A duração total de uma fornada, incluída sua preparação, isto é, o tempo gasto com o carregamento e descarregamento do forno, varia conforme o grupo de ligas da fornada, estando entre oito e nove horas. Tal duração depende apenas do grupo de ligas da própria fornada, não dependendo significativamente da quantidade de aço no forno.

Para o processamento dos aços considerados existe apenas um forno, funcionando em três turnos.

O desbaste é realizado em um laminador dedicado, onde não é necessário preparação em função do tipo de produto. Sua produtividade varia também de acordo com o grupo de ligas utilizado, indo de 9 a 10,5 toneladas por hora.

O laminador existente, também único para a laminação final dos perfis, é preparado em função da faixa de bitola a ser processada. Conforme já explicado, de

acordo com as dimensões e com o tipo de perfil desejado pelo cliente, pode-se atribuir a um item de pedido, uma faixa de bitola. Existem oito montagens diferentes para o laminador considerado, uma para cada faixa de bitola.

Uma montagem, ou preparação do laminador, pode durar de 4 a 15 horas, de acordo com a faixa de bitola para a qual a preparação está sendo realizada. Da mesma forma, a produtividade deste laminador também depende da faixa de bitola processada, podendo ir de 1,47 a 2,31 toneladas por hora. Como o forno, os dois laminadores também trabalham em três turnos.

As operações de acabamento podem ser de trefilação, retífica, torneamento ou fresamento. Os produtos podem também ser entregues brutos. A empresa dispõe dos valores de *lead times* para cada tipo de acabamento, variando também em função do grupo de liga.

1.5 O PROBLEMA DA PROGRAMAÇÃO DA PRODUÇÃO

Dada uma carteira de pedidos, o problema consiste em decidir a sequência de processamento das ordens de produção, geradas por cada um dos itens dos pedidos, em cada etapa do processo ~~de maneira ótima~~. Esta otimização deve levar em conta vários critérios de desempenho, normalmente conflitantes entre si:

- **ocupação e utilização dos equipamentos,**
- **nível dos estoques,**
- **número de preparações e**
- **atendimentos aos prazos fixados**

Para ilustrar melhor quão complexo é o problema, sejam consideradas apenas a fundição e a laminação. Na primeira, o processamento é feito por liga, ou seja, ordens de produção de uma mesma liga devem compor uma fornada. Já na laminação, tal composição deve ser feita por faixa de bitola, para que o tempo de preparação do laminador, que é elevado, seja diluído no lote. Quanto maior o número de ordens de produção de uma mesma faixa de bitola processados em sequência, maior a diluição do tempo de preparação e melhor a utilização do laminador.

Olhando o problema apenas sob este ponto de vista, uma boa maneira de ocupar bem os dois equipamentos seria:

- ordenar a lista de ordens de produção por liga e, entre as ordens de uma mesma liga, ordená-los por faixa de bitola e
- compor as fornadas a partir desta lista ordenada, atribuindo sequencialmente cada ordem a uma fornada até ocupá-la ao máximo e assim sucessivamente, respeitando as restrições de somente uma liga numa fornada e de não haverem fornadas com menos de 10 t.

Com as fornadas compostas seria necessário decidir a ordem de processamento destas. Para cada fornada poder-se-ia encontrar a faixa de bitola predominante na mesma e processá-las em ordem de faixa de bitola predominante. Isso faria com que os itens saíssem da fundição razoavelmente em ordem de faixa de bitola, possibilitando uma sequência de processamento para o laminador consideravelmente boa.

Com essa heurística seria maximizada a ocupação do forno e criada uma tendência de boa utilização do laminador, já que as fornadas com mesma faixa de bitola predominante seriam processadas em sequência. Um baixo número de preparações e bom nível de estoque intermediário (tarugos fundidos e não laminados) também seriam favorecidos. Mas sabe-se lá o que aconteceria com o atendimento aos prazos.

Conforme ilustrado, a solução com uma regra heurística, irá, na melhor das hipóteses, desconsiderar parte dos objetivos para privilegiar outros.

Ao invés de se utilizar uma regra heurística, poder-se-ia, então, utilizar um método de busca para se obter a sequênciação ótima, ou próxima disso, com avaliação quantitativa das soluções. Neste caso, a sequenciação a ser feita poderia envolver mais de 1000 ordens de produção, resultando num total de $(1000!)^2$ possibilidades de sequenciações diferentes, considerando-se apenas a fundição e a laminação. A utilização de um método de busca num espaço de soluções tão vasto seria inviável mesmo em computadores de grande porte.

Observou-se, então, a necessidade de se utilizar um método de busca combinado com regras heurísticas que limitassem o número de soluções a serem avaliadas pelo método, restringindo a busca a espaços com soluções mais promissoras.

2.

FUNDAMENTOS TEÓRICOS

2.1 INTRODUÇÃO[1]

O Simulated Annealing é um método para obtenção de boas soluções para problemas complexos de otimização que tem recebido muita atenção nos últimos anos. Este interesse começou com Kirkpatrick (1983) e Cerny (1985), que mostraram que o modelo para simulação do recozimento de materiais proposto por Metropolis et al. (1953) poderia ser utilizado em problemas de otimização.

A partir de então, o Simulated Annealing tem sido aplicado a muitos problemas de otimização de diversas áreas como processamento de imagens, física e química molecular e programação da produção. Também houveram progressos advindos da análise matemática do método, bem como diversas experiências em computador comparando Simulated Annealing com outros métodos.

2.2 DESCRIÇÃO DO SIMULATED ANNEALING

O Simulated Annealing é um tipo de método de busca. Uma forma simples de método de busca começa com uma solução inicial que pode ser gerada aleatoriamente. Uma solução vizinha a essa é então gerada por algum mecanismo apropriado e a mudança no custo é calculada. Se uma redução de custo é encontrada, a solução vizinha toma o lugar da solução atual, caso contrário, a solução atual é mantida. O processo é repetido até que nenhuma redução de custo possa ser obtida na vizinhança da solução atual e o algoritmo é terminando num mínimo local.

Embora essa forma seja simples e de rápida execução, a desvantagem é que o mínimo local encontrado pode estar bastante longe do mínimo global. Uma maneira melhorar tal método é executá-lo várias vezes, começando de soluções iniciais diferentes e selecionar o melhor dos mínimos locais encontrados. Ao invés dessa estratégia, o Simulated Annealing tenta evitar ficar preso num mínimo local aceitando, às vezes, uma mudança para uma solução vizinha que aumente o valor do custo. A aceitação ou rejeição de uma mudança para uma solução de maior custo é determinada por uma sequência de números aleatórios, mas com uma probabilidade controlada. A probabilidade de aceitar uma mudança que cause um aumento δ no custo é chamada de

[1] Adaptado de EGLESE (1990).

função de aceitação e é normalmente determinada pela função exp(-δ / T) onde T é um parâmetro de controle que corresponde à temperatura numa analogia ao recozimento de metais. Isto implica que pequenos incrementos no custo são mais aceitáveis do que grandes incrementos e que quando T é alto mais mudanças ocorrerão, mas quando T se aproxima de zero a maioria das mudanças com aumento de custo serão rejeitadas.

Assim, no Simulated Annealing, o algoritmo é iniciado com um valor de T relativamente alto, para evitar que se caia prematuramente num ótimo local. São realizados um certo número de tentativas de mudança a cada temperatura, enquanto este parâmetro é gradualmente reduzido. O algoritmo do Simulated Annealing é ilustrado em pseudocódigo na tabela 2.1.

Tabela 2.1: Algoritmo do Simulated Annealing em pseudocódigo.

```
Fixa um estado inicial i;
Fixa uma temperatura inicial T > 0;
Inicializa contador de mudança de temperatura t = 0;
Repeat
  Inicializa contador de repetição n = 0;
  Repeat
    Gera um estado j, vizinho de i;
    Calcula δ = f(j) - f(i);
    If δ < 0 then i := j
    else if random(0,1) < exp(-δ / T) then i := j;
    n := n + 1;
  until n = N(t);
  t := t +1;
  T := T(t);
until critério de parada verdadeiro.
```

2.3 ANALOGIA FÍSICA

A motivação para o Simulated Annealing vem da analogia entre o recozimento[2] de materiais sólidos e problemas combinatórios. No recozimento, são buscados estados de baixa energia em um material fundindo-se o mesmo e então diminuindo-se a temperatura lentamente até a solidificação completa. Num líquido, as partículas são arranjadas aleatoriamente. Mas, num sólido cristalino, elas terão uma estrutura particular

[2] Tradução de annealing.

correspondendo a uma configuração de mínima energia. Se o resfriamento não for feito lentamente, o sólido não irá atingir tal estado e sim um estado parcialmente estável, localmente mínimo, como vidro ou um cristal com vários defeitos na estrutura.

As diferentes estruturas da substância correspondem às diferentes soluções de um problema combinatório de otimização e a energia do sistema corresponde a função a ser minimizada. O método de busca simples descrito anteriormente corresponde ao rápido resfriamento, onde apenas mudanças que resultam na redução da energia do sistema são aceitas. A determinação da temperatura inicial, da taxa na qual ela é reduzida, do número de iterações a cada temperatura e do critério de parada da busca é chamada de tabela de resfriamento. Essa escolha tem importantes efeitos na performance do algoritmo.

2.4 CONCLUSÕES TEÓRICAS

O algoritmo do Simulated Annealing pode ser modelado utilizando-se a teoria das cadeias de Markov. Fixando-se o parâmetro de temperatura T, a matriz de transição P_{ij}, que representa a probabilidade de mudança do estado i para o estado j, é independente do número de iterações, o que corresponde a uma cadeia de Markov homogênea. Pode ser demonstrado que, garantindo-se que seja possível chegar a qualquer estado j a partir de qualquer estado i em número finito de mudanças com probabilidade diferente de zero, então a cadeia de Markov correspondente ao algoritmo do Simulated Annealing tem uma distribuição de regime única q(i), que não depende do estado inicial. Como corolário deste resultado, o limite quando $T \rightarrow 0$ dessa distribuição de regime é uma distribuição uniforme sobre o conjunto de soluções ótimas globais, isto é, o Simulated Annealing converge assintoticamente para o conjunto de soluções ótimas globais.

A convergência assintótica para o conjunto de soluções ótimas globais é uma conclusão encorajadora e que permanece para vários tipos de funções de aceitação e de mecanismos de mudança, mas não diz nada a respeito sobre o número de iterações necessárias para se chegar à convergência. Aarts e van Laarhoven (1985) mostrou que pode-se chegar perto desta distribuição de regime apenas se o número de iterações é pelo menos o quadrado do tamanho do espaço de soluções. Uma vez que o tamanho do espaço de soluções é frequentemente exponencial em relação ao tamanho do problema, é

necessário um tempo de execução também exponencial para se chegar a esta distribuição.

Além de poder ser descrito como uma sequência de cadeias de Markov homogêneas de comprimento finito, o algoritmo do Simulated Annealing também pode ser considerado uma única cadeia de Markov não homogênea com as probabilidades de transição dependentes do número de iterações. Usando esse novo modelo, vários métodos foram obtidos para dar condições para a convergência para o conjunto de soluções ótimas globais. Hajek (1988) obteve o melhor resultado no qual estabeleceu condições necessárias e suficientes para isto. Ele mostrou que se $T(k) = c / \log(1 + k)$ onde k é o número da iteração, a convergência assintótica é garantida se c for maior ou igual a profundidade do mais profundo mínimo local que não é um mínimo global. Esta função de temperatura representa um resfriamento muito lento. Também pode ser demonstrado (e.g. Mitra et al., 1986) que a tentativa de se chegar arbitrariamente perto da distribuição uniforme sobre o conjunto de soluções ótimas leva a um número de iterações maior que o tamanho do espaço de soluções do problema, resultando, assim, num tempo de execução exponencial em relação ao seu tamanho. Note-se que o resultado teórico para convergência assintótica para do Simulated Annealing modelado como uma cadeia de Markov não homogênea não necessita da existência de distribuições de regime para valores de T fixos.

2.5 IMPLEMENTAÇÃO DO SIMULATED ANNEALING

Para se usar o Simulated Annealing num particular problema de otimização é necessário fazer uma série de escolhas. Tais escolhas podem ser divididas em dois grupos: escolhas específicas ao problema e escolhas genéricas.

2.5.1 ESCOLHAS ESPECÍFICAS AO PROBLEMA

O problema precisa ser claramente formulado, de forma que o conjunto de soluções viáveis seja definido. O método de obtenção das soluções vizinhas precisa também ser definido, assim como a maneira de se determinar o valor do objetivo a ser minimizado (ou a mudança neste valor). Uma solução inicial precisa ser gerada.

Vários pesquisadores (e.g. Cerny, 1985; Matsuo et al., 1988) mostraram que a eficiência do Simulated Annealing depende da estrutura de vizinhança usada. Isso explica o resultado obtido por Hajek apresentado no item anterior, mostrando que a convergência assintótica para o conjunto de soluções ótimas depende da profundidade do mínimo local. Em geral, uma estrutura de vizinhança que impõe uma topologia suave, onde os mínimos locais atingidos são pouco profundos é preferível a uma topologia acidentada, onde a execução do problema passa por muitos mínimos profundos.

Uma outra escolha precisa ser feita para problemas com restrições. É necessário decidir se o espaço de soluções será restringido ou se serão permitidas soluções que não respeitem as restrições mas com uma função objetivo que penalize tais violações.

2.5.2 ESCOLHAS GENÉRICAS

As escolhas a serem feitas, e que constituem a tabela de resfriamento, são:

- o valor inicial do parâmetro T,
- a função temperatura, T(t), que determina como a temperatura será mudada,
- o número de iterações, N(t), a ser realizado a cada temperatura, e
- o critério de parada para encerrar o algoritmo.

Uma grande variedade de tabelas de resfriamento já foram sugeridas. A primeira delas era baseada na analogia com o recozimento de sólidos. Assim, por exemplo, Kirkpatrick et al. (1983) fixa o valor inicial de T alto o suficiente para que virtualmente todas as mudanças fossem aceitas, o que correspondia ao aquecimento de uma substância até que todas as partículas estejam randomicamente arranjadas como num líquido. Uma função de temperatura proporcional é usada, isto é, $T(t + 1) = \alpha T(t)$ onde α é constante. Valores típicos de α usados na prática estão entre 0,8 e 0,99. Tal função propicia decrementos menores em T quando este está próximo de zero. O número de iterações em cada valor de T, N(t), é determinado e um limite máximo de mudanças é fixado. Assim, tenta-se passar para o sistema o estado correspondente ao equilíbrio térmico na analogia física. O algoritmo é interrompido quando a solução obtida permanece a mesma por um certo número de passos de temperatura. O estado final corresponde à solidificação.

Outras tabelas de resfriamento são mais diretamente orientadas pelas conclusões teóricas sobre convergência assintótica descritas no item anterior. N(t) é definido de

forma que o sistema fique suficientemente perto da distribuição de regime para todos os valores de T. Aarts e Korst (1989), e van Laarhoven e Aarts (1987) referem-se a isto como um estado de 'quase-equilíbrio'. Diferentes regras ou heurísticas para decidir quando o 'quase-equilíbrio' é atingido leva a diferentes tabelas de resfriamento.

Contudo, como as conclusões de Hajek (1988) mostram, para a convergência assintótica para o conjunto de soluções ótimas, a condição necessária é que o resfriamento ocorra lentamente, e não que um estado correspondente ao equilíbrio térmico seja atingido. Assim, num extremo há, por exemplo, o esquema proposto por Lundy e Mees (1986), no qual há apenas uma iteração a cada temperatura. Eles usam argumentos heurísticos para obter uma função de temperatura na forma $T(t + 1) = T(t) / (1 + BT(t))$, onde B é uma constante, sendo tal função equivalente a $T(t) = C_1 / (1 + tC_2)$ onde C_1 e C_2 são constantes. Com um grande número de iterações, isto irá representar resfriamento mais lento do que uma redução proporcional de temperatura com α e N fixos. No outro extremo há o esquema heurístico baseado no Simulated Annealing proposto por Connolly (1988) no qual a maioria das iterações deve ser feita a uma mesma temperatura.

Uma distinção que deve ser feita entre as diferentes tabelas de resfriamento é a existência de tabelas com e sem *feedback* para T(t) e N(t). Tabelas simples sem *feedback* determinam N(t) e T(t) no começo da execução. Outras tabelas permitem que *feedbacks* no decorrer do algoritmo afetem os valores de tais funções. A tabela descrita por Kirkpatrick et al. (1983), apresentada anteriormente, permite que o número de mudanças aceitas afete N(t).

2.6 OBJETIVO DO TRABALHO

O objetivo do trabalho foi estudar a utilização do método de busca Simulated Annealing na programação da produção de perfis de aço laminados. A partir de um modelo do sistema, foi elaborado um programa de computador, que doravante será chamado de Sequenciador, que propõe, para as principais etapas do processo, listas que determinam a sequência de produção das diversas ordens.

Avaliando estas soluções através de uma função de custo e utilizando o Simulated Annealing, o programa propõe uma boa solução para a programação da produção.

Partindo de uma solução inicial baseada em uma regra heurística que fornece uma solução razoável, avaliou-se a capacidade do programa em buscar soluções melhores que ela, considerando-se também o tempo de processamento gasto para tanto.

3.

MODELAGEM DO PROBLEMA

3.1 INTRODUÇÃO

Várias simplificações foram adotadas na elaboração do modelo do sistema produtivo, visando tanto a viabilidade da execução do Sequenciador, relacionada ao tempo de processamento e ao número de alternativas de sequenciações geradas, quanto a possibilidade de se implementá-lo no período de tempo disponível para a realização do trabalho.

Assim, além das simplificações citadas no primeiro capítulo, ligadas aos roteiros de produção considerados, pode-se observar outras ao longo deste capítulo. Essas simplificações, porém, foram feitas em aspectos secundários do sistema, de forma a não comprometer as decisões principais do ponto de vista da programação da produção.

Ao final do trabalho, seus resultados indicarão se mais características do processo produtivo real poderão ser incorporadas ao programa ou se isto é inviável.

3.2 CONSIDERAÇÕES GERAIS

Para este problema foi utilizado um modelo de simulação estático, isto é, foi obtido um número fixo de ordens de produção para que o programa realizasse a sequenciação das mesmas.

Assim, não foram consideradas as ordens de produção que já estivessem em máquina. O Sequenciador considera que as máquinas estão paradas. Por outro lado, ele também não considera a possibilidade de chegarem novos pedidos durante a produção e estes entrarem na sequenciação.

Os tempos de preparação e de produção foram considerados determinísticos e as máquinas disponíveis todo o tempo (vale lembrar que estas trabalham em três turnos).

3.3 SEQUENCIAÇÕES CONSIDERADAS

Das quatro etapas do processo (fundição, desbaste, laminação e acabamento), a operação de desbaste é a mais simples. Nela não há necessidade de preparação de máquina como na laminação e possui uma produtividade muito superior à fundição e à laminação. Dessa forma, julgou-se razoável a desconsideração de uma nova sequenciação das ordens antes desta etapa. As ordens serão processadas de acordo com

sua saída da fundição, não sendo considerada também a restrição de capacidade da máquina.

No acabamento, onde existe um grande número de máquinas diferentes e também não há preparações de grande duração, a sequênciação das ordens será a mesma da laminação.

3.4 DEFINIÇÃO DA SOLUÇÃO

Assim, uma solução para a programação da produção deverá definir a sequência em que as ordens entrarão na fundição e a sequência em que as ordens entrarão na laminação. Desta forma, estarão sendo tomadas as decisões mais difíceis, e com maior potencial de impacto nos resultados.

Mesmo sendo restrita a duas sequenciações, a resolução do problema com o número de ordens de produção perto de mil resulta num número descomunal de possibilidades. Isso se as ordens forem apenas sequenciadas, e não se considerar a possibilidade de inserção de esperas entre o processamento delas, o que poderia produzir soluções melhores.

Tornou-se necessário, então, restringir este espaço de soluções através de uma regra heurística que descartasse soluções obviamente ruins, deixando ainda uma grande quantidade de possibilidades para resolução através do método de busca.

3.5 SOLUÇÃO PARA A FUNDIÇÃO

Dado que na fundição o processamento é feito por lotes, decidiu-se realizar a composição dos lotes, ou fornadas, por uma regra heurística, cabendo ao Simulated Annealing decidir apenas a sequência de processamento das fornadas.

As fornadas serão formadas da seguinte maneira: para cada liga a ser processada[1], deve ser feita uma lista de ordens de produção pertencentes a ela, sendo ordenada por folga dinâmica[2]. Com a lista nesta ordem, irão sendo compostas as fornadas, de forma que sua ocupação seja máxima. Caso a última fornada fique com

---

[1] Vale lembrar que uma fornada deve ter ordens pertencentes a uma mesma liga.

[2] Seu cálculo é feito subtraindo-se os tempos de fundição, desbaste, laminação e lead time do acabamento do prazo de entrega.

menos de 10 t, que é o limite mínimo do forno, ordens da fornada imediatamente anterior podem ser passadas para ela[3].

Assim, as ordens mais atrasadas de cada liga estarão todas em uma mesma fornada. Mais do que isso, todas as fornadas estarão cheias com ordens com folga dinâmica semelhante.

Fica estabelecida, então, a solução para a fundição: o Sequenciador deve gerar uma sequência de fornadas, com as ordens pertencentes a cada uma definida através da regra apresentada.

Com esta definição privilegiou-se fortemente a ocupação e utilização das fornadas, já que elas são definidas com ocupação máxima e não são colocadas esperas entre o processamento de uma e outra[4]. Em segundo lugar, o atendimento aos prazos foi beneficiado, já que as fornadas mais atrasadas podem ser processadas em primeiro lugar[5]. Outros critérios como nível do estoque intermediário (tarugos fundidos e não laminados), utilização e número de preparações do laminador podem ser prejudicados.

De uma forma geral, esta regra deve ser feita levando-se em consideração os critérios de desempenho mais importantes para a empresa, de forma que sejam prioritariamente beneficiados.

3.6 SOLUÇÃO PARA LAMINAÇÃO

Uma regra também foi definida para a laminação, de forma a restringir o espaço de soluções a aquelas mais promissoras.

Pode-se dizer que, quando um lote é fundido, as ordens, após o desbaste, vão sendo posicionadas em filas imaginárias, uma para cada faixa de bitola, e ficam aguardando o laminador ser preparado para a processamento da sua fila.

Como regra heurística, estabeleceu-se que cada uma destas filas deve ser processada até que não haja mais nenhuma ordem nela. Acabando a fila, o laminador

---

[3] Tal regra define apenas quais ordens pertencerão a quais fornadas, mas não a sequência de produção das fornadas.

[4] Tal procedimento poderia ser utilizado para se obter menores níveis de estoque intermediário.

[5] Pode-se falar em 'fornadas mais atrasadas', pois a heurística produz fornadas com ordens com atrasos próximos uns dos outros.

deve ser preparado para processar outra fila, podendo, mais tarde, voltar a ser preparado para a primeira caso ainda tenham sobrado ordens desta faixa de bitola na fundição.

Havendo várias ordens na fila, deverá ser processada primeiro aquela com menor folga dinâmica, calculada subtraindo-se os tempos de laminação e o *lead time* do acabamento do prazo de entrega.

Define-se, então, a regra de formação dos lotes de laminação: ao se começar processar uma faixa de bitola, deve-se manter o laminador produzindo ordens desta faixa até que não haja mais nenhuma disponível para laminar, devendo o processamento desse lote ser feito por ordem de folga dinâmica.

Pode-se agora definir a solução para a laminação: fixando-se uma lista de prioridades que defina para qual faixa de bitola o laminador deve ser preparado ao final de um lote, está definida a solução, uma vez que a regra estabelecida define a sequência em que as ordens serão processadas e o evento para a mudança para a próxima faixa de bitola da lista. De forma semelhante ao que foi definido para a fundição, ao Simulated Annealing caberá apenas a definição da lista de faixas de bitola.

O fluxograma a seguir mostra o funcionamento deste método:

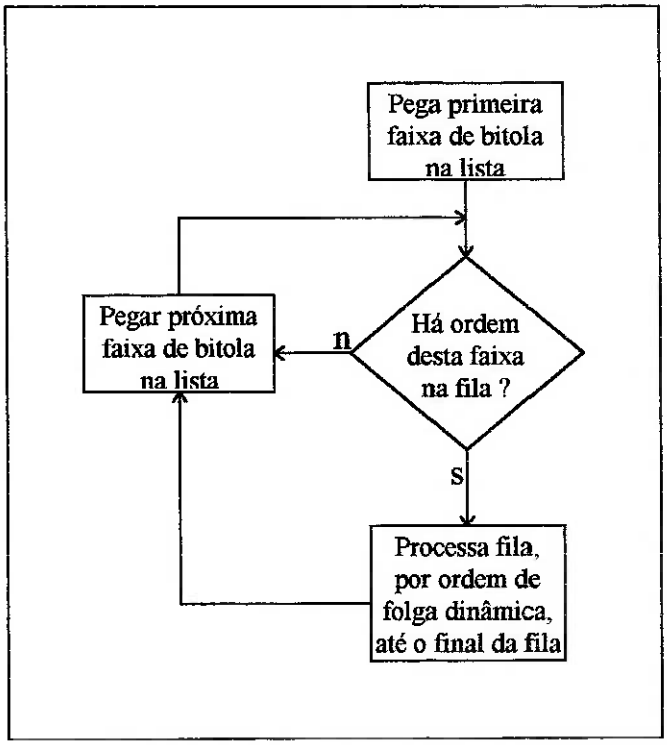

Figura 3.1: Fluxograma da solução para a laminação.

Ao se chegar ao final da lista de faixas de bitola, certamente ainda não devem ter sido processadas todas as ordens, pois algumas ainda estarão sendo fundidas ou desbastadas ao final do processamento do primeiro lote da sua faixa de bitola. Deve-se, então, recomeçar a lista novamente, tantas vezes quantas forem necessárias.

Com essa definição de solução, tentou-se melhorar o número de preparações do laminador, não interrompendo o processamento de uma faixa de bitola enquanto não tiverem sido processadas todas as ordens dela já desbastadas. Com a solução definida para a fundição, este critério não havia sido muito beneficiado, pois a formação das fornadas não leva em conta a que faixa de bitola a ordem pertence. A regra da fundição não se importa se as ordens de uma mesma faixa estão distribuídas pelas diferentes fornadas, ao invés de estarem concentradas em poucas, o que melhoraria a utilização da laminação.

Esta solução da laminação melhora um pouco o desempenho neste critério, baixando também o nível do estoque intermediário, já que o laminador terá boa produtividade.

Conforme já citado, essas propostas de solução beneficiam prioritariamente critérios como utilização e ocupação da fundição e em segundo lugar prazos de entrega, ficando por último os estoques e o número de preparações do laminador. Outras regras poderiam ser concebidas de forma a privilegiar os critérios de desempenho mais importantes para a empresa.

Tanto na fundição quanto na laminação, a folga dinâmica foi utilizada como critério de ordenação das ordens de produção. Na fundição, ela serve de critério para se decidir quais ordens vão para cada fornada. Na laminação, define a ordem de processamento das ordens de produção de um mesmo lote.

Uma vez que a importância ao atendimento ao prazo de entrega varia de acordo com o tamanho do pedido, tal folga deve ainda ser multiplicada pela quantidade desejada pelo cliente, para que a ordenação seja realmente justa. Assim, um pedido de 1000 toneladas que tem dez dias de folga deve ser processado primeiro que um pedido de 500 toneladas com nove dias de folga, de forma a minimizar-se dias de folga vezes quantidade.

O fato de o modelo para o problema ser estático e considerar que os equipamentos não estão em produção no início da simulação poderia acarretar uma distorção na formação dos lotes da laminação. Ao chegarem nela as ordens da primeira

fornada, formariam pequenas filas imaginárias, conforme descrito anteriormente. Uma vez que a regra heurística adotada para o laminador determina que tais filas sejam processadas até o final e em seguida o laminador seja preparado para outra fila, a existência de filas muito curtas poderia provocar um número exagerado de trocas, geradas por uma simplificação do modelo.

Para evitar tal problema, foi definido um parâmetro chamado espera, que determina o tempo que o laminador deve ficar parado no início da simulação, até que as filas por faixa de bitola fiquem com o tamanho adequado.

3.7 TEMPOS DE PRODUÇÃO

Os tempos de produção das ordens em cada etapa do processo foram considerados determinísticos, mas com variações em função de características do produto como grupo de liga e faixa de bitola a que ele pertence, além do seu acabamento. A tabela abaixo mostra esta dependência, e o número de grupos, faixas de bitola ou acabamentos, que determinam o número de valores diferentes de cada tempo.

Tabela 3.1: Dependência entre tempos de produção e características dos produtos.

Tempo	**Característica**	**Variações**	**Faixa**
fundição	grupo de liga	5	8 a 9 h por fornada
desbaste	grupo de liga	5	9 a 10,5 t por hora
laminação - preparação	faixa de bitola	8	4 a 15 h
laminação - processamento	faixa de bitola	8	1,47 a 2,31 t
acabamento	acabamento	6	16 a 72 h
	grupo de liga	5	

3.8 AVALIAÇÃO DE DESEMPENHO

Definido o que será uma solução para a fundição e para a laminação, é necessário traduzir-se os critérios de desempenho em termos quantificáveis, de forma que soluções diferentes possam ser comparadas por um programa de computador. O desempenho de uma solução em relação a critérios distintos como estoques, cumprimento de prazos, número de preparações e ocupação e utilização dos equipamentos precisam ser medidos e somados, resultando numa cifra única de desempenho.

Assim, essa medição está dividida em duas etapas. A primeira refere-se à medição 'física' de cada um dos objetivos: qual a quantidade atraso ou adiantamento ocorrido, a quantidade de estoque intermediário, quantas preparações foram executadas e quão bem os equipamentos foram utilizados e ocupados. A seguir, essas quantidades, em unidades diferentes uma das outras, devem ser ponderadas de acordo com sua importância e transformadas para uma unidade comum de desempenho, que, no caso, é a unidade monetária: a quantidade de atraso será multiplicada por uma penalidade financeira por dia de atraso, o mesmo ocorrendo com o adiantamento e assim por diante. Somando-se as diversas parcelas ter-se-á a função custo, a qual será minimizada pelo Simulated Annealing.

São definidas, a seguir, a medida física e a transformação de unidade para cada um dos objetivos.

Atrasos ou adiantamentos

Pode-se considerar que o descontentamento dos clientes da empresa varia linearmente com o número de dias de atraso. Para simplificar a modelagem, considerou-se que o descontentamento só varia com o número de dias de atraso e com a quantidade, não dependendo do cliente, nem de outra característica. Assim, a medida definida é a somatória do número de dias de atraso multiplicado pela quantidade de cada uma das ordens.

O término de um pedido antes da data solicitada pelo cliente também deve ser medido e penalizado, já que, face a dificuldade de faturamento e entrega neste caso, o produto acabado necessita ficar estocado na empresa, gerando de despesas de armazenagem e financeiras. Para sua medição, decidiu-se utilizar o mesmo critério do atraso.

Em seguida, as duas quantidades serão multiplicadas por suas respectivas penalidades financeiras por dia vezes tonelada de atraso ou adiantamento.

Estoque intermediário

Para a medição do desempenho em relação ao estoque intermediário, será obtido o número de dias que cada ordem ficou no estoque intermediário, isto é, qual o tempo decorrido entre sua saída do desbaste e o começo de sua laminação. Tal número será

multiplicado pela quantidade de material e será obtido um total para todas as ordens. Assim, ter-se-á a medida física do estoque em toneladas vezes dias. Para a conversão para unidade monetária, o total será multiplicado por uma taxa por dia vezes tonelada de estoque.

Nesta taxa, estão embutidos o custo de armazenagem propriamente dito e o custo financeiro do estoque.

A rigor, o custo financeiro do estoque deveria ser calculado com base no valor estocado, e não na quantidade. Tal procedimento é uma aproximação.

Preparações

A medição das preparações do laminador será simplesmente por contagem, sendo em seguida multiplicada pelo custo unitário.

Utilização e ocupação dos equipamentos

A medição da ocupação dos equipamento foi considerada desnecessária. Na fundição, a regra estabelecida determinou a ocupação máxima do forno em cada processamento e esta não varia de solução para solução, já que as fornadas serão sempre as mesmas. Uma vez que o desempenho em relação a este critério não mudaria, não faria sentido incorporá-lo numa avaliação comparativa. No laminador, não faz sentido falar-se em ocupação.

A medição da utilização dos equipamentos será feita através do cálculo do tempo total de operação do sistema gasto para o processamento de todas as ordens, o qual será chamado de horizonte de planejamento.

Para facilitar a atribuição de valor à taxa monetária, a medição leva em conta o desempenho ideal neste critério, num problema estático. Assim, a taxa monetária, em unidades monetárias por dia, multiplicará o horizonte de planejamento para a dada solução subtraído do horizonte de planejamento ideal.

Faz-se necessário, desta forma, definir tal desempenho. Para a fundição, a utilização ideal é conseguida colocando-se o maior número de ordens em cada fornada e processando-as ininterruptamente. Pode-se, então, calcular o horizonte para a fundição das ordens nesta sequência: é o tempo decorrido desde o início da fundição da primeira fornada até o final da fundição da última, mais os tempos processamento no desbaste, na

laminação e o *lead time* do acabamento para a última ordem de produção fundida. Este período de tempo será chamado de horizonte de fundição ideal.

Da mesma forma, o processamento de todas as ordens de produção de cada faixa de bitola juntas e ininterruptamente proporciona um número mínimo de preparações. Isso é possível se se considerar que uma ordem já estará fundida quando chegar sua vez de ser laminada (suponha-se que tenham sido fundidas numa sequência que proporcione tal fato ou que todas as fundições estejam terminadas). É possível, assim, calcular o horizonte ideal para a laminação: é o tempo decorrido desde o início da preparação para a primeira faixa de bitola até o final da laminação da última ordem da última faixa, mais os tempos de fundição e desbaste da primeira ordem de produção laminada e o tempo de acabamento da última.

A medida deste critério é definida como o número de dias de funcionamento do sistema necessário para o processamento de todos os itens na dada sequência menos o maior dos dois horizontes ideais definidos acima.

Apresenta-se agora um tabela que resume as medidas de desempenho em relação a cada um dos critérios adotados.

Tabela 3.2: Medidas de desempenho.

Critério	**Medida**	**Taxa**
atraso	$\sum_{\substack{i \\ \text{(ordens atrasadas)}}} (\text{data de término}_i - \text{data de entrega}_i).\text{quantidade}_i$	\$/t.dia
adiantamento	$\sum_{\substack{i \\ \text{(ordens adiantadas)}}} (\text{data de entrega}_i - \text{data de término}_i).\text{quantidade}_i$	\$/t.dia
estoque intermediário	$\sum_{i} (\text{início da laminação}_i - \text{final do desbaste}_i).\text{quantidade}_i$	\$/t.dia
preparações da laminação	número de preparações	\$/preparação
utilização dos equipamentos	horizonte de planejamento da solução - maior(horizonte ideal para fundição, horizonte ideal para laminação)	\$/dia

4.

O SIMULATED ANNEALING

4.1 INTRODUÇÃO

Além da definição da solução propriamente dita, a implementação do Simulated Annealing exige que se elabore um método de geração de uma nova solução, ou solução vizinha, a partir de uma solução existente. É necessário também que se estabeleça como a primeira solução, também chamada solução semente, será gerada. Por fim, a chamada tabela de resfriamento precisa ser montada, determinando vários parâmetros como temperatura inicial, função de temperatura, número de iterações a cada passo de temperatura e o critério de parada do algoritmo.

4.2 GERAÇÃO DE SOLUÇÃO VIZINHA

Uma opção de se estruturar a forma de obtenção da solução vizinha seria a geração de uma solução completa, isto é, solução para fundição mais solução para laminação, a partir de outra solução completa. Poder-se-ia, por exemplo, sortear-se duas fornadas e trocá-las de posição na ordem de processamento, trocando-se também duas faixas de bitola na lista de prioridades de faixa. Assim, de uma solução completa obter-se-ia outra solução completa.

Esse modo de geração teria o inconveniente de não considerar a dependência entre as duas partes da solução. Uma solução boa na laminação depende da sequência em que as ordens de produção irão chegar nela. Assim, uma boa solução para a laminação deveria ser buscada com a sequência de fundição fixa.

Decidiu-se, então, estruturar a geração de vizinhos num duplo simulated annealing: um somente da laminação e outro completo, que inclui também o primeiro.

Com uma solução fixada para fundição, é realizado um simulated annealing na laminação para encontrar a solução para a laminação, isto é, a lista de faixas de bitola, que melhor se adapte a solução da fundição, gerando menor custo. Encontrada esta solução, gera-se outra solução para fundição e repete-se o simulated annealing da laminação. É obtida, então, a melhor solução da laminação para esta segunda solução da fundição. As duas soluções são comparadas e a melhor é guardada. Este é o simulated annealing completo, algumas vezes também chamado de simulated annealing da fundição.

A seguir, o fluxograma ilustra a estrutura de duplo simulated annealing.

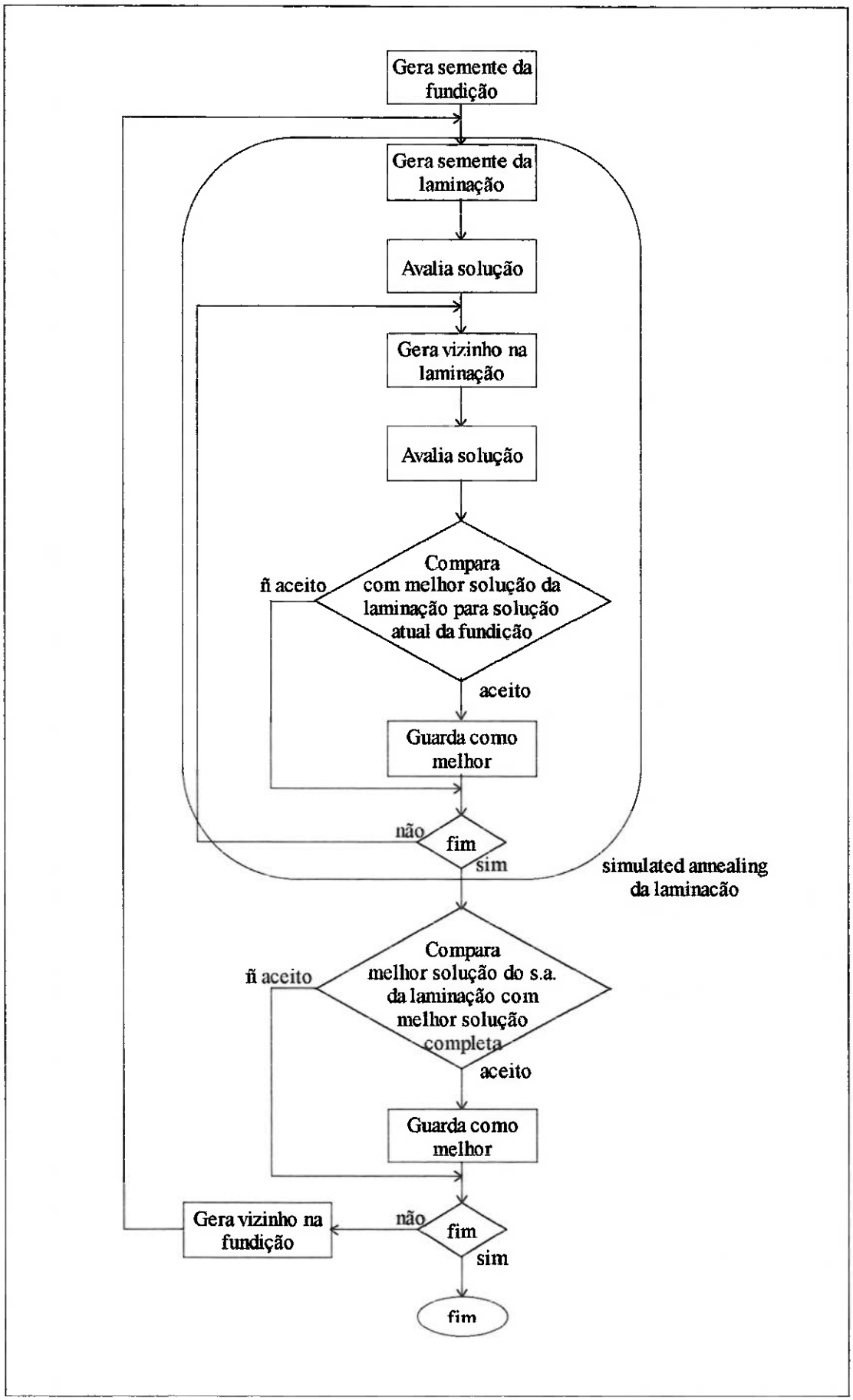

Figura 4.1: Fluxograma do duplo simulated annealing.

Esta estrutura, ao contrário da anterior, considera a dependência entre a solução da laminação e a solução da fundição na busca do menor custo.

4.2.1 NA FUNDIÇÃO

Dado que na fundição as fornadas serão definidas no início do processamento, através da regra heurística descrita, a geração de soluções diferentes será feita através de alteração na ordem de processamento dessas fornadas.

Tendo-se uma lista de fornadas a serem processadas como uma solução, a solução vizinha será obtida sorteando-se duas fornadas da lista e invertendo-se a ordem de processamento das fornadas situadas entre elas, conforme a figura abaixo.

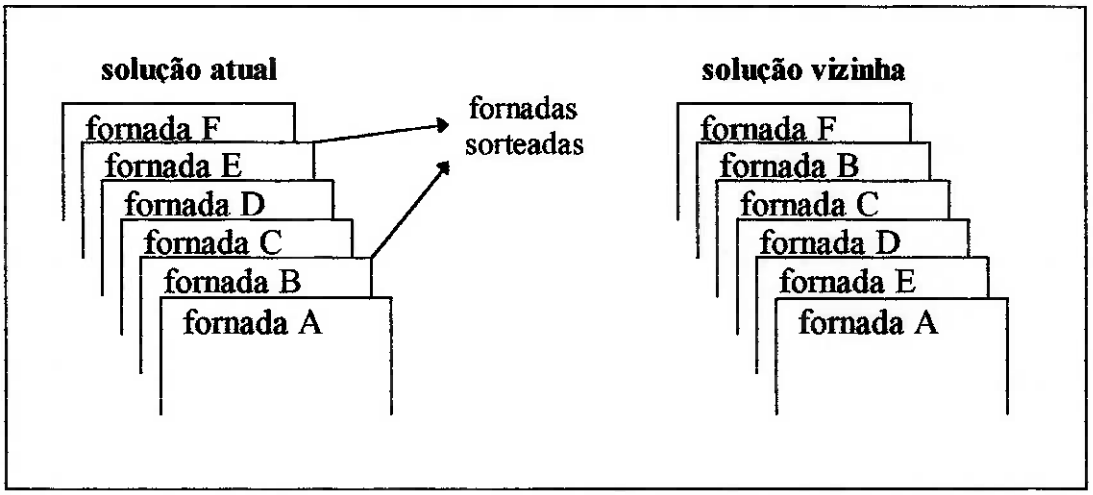

Figura 4.2: Solução vizinha na fundição.

Se as duas fornadas que definem o trecho de fornadas a ser invertido fossem sorteadas livremente, a solução vizinha obtida poderia ser muito diferente da solução fonte. Conforme explicado no capítulo 2, a eficiência do Simulated Annealing depende da estrutura de vizinhança usada. Em geral, uma estrutura de vizinhança que impõe uma topologia suave, onde os mínimos locais atingidos são pouco profundos é preferível a uma topologia acidentada, onde a execução do problema passa por muitos mínimos profundos. Assim, limitou-se o tamanho do trecho a ser invertido, fazendo com que a solução gerada seja realmente vizinha da solução atual, favorecendo a imposição de uma topologia suave.

Vale lembrar que a solução vizinha é sempre gerada a partir da última solução aceita, isto é, se uma solução é gerada, avaliada e descartada, por exemplo, por

apresentar um acréscimo grande de custo, a próxima solução vizinha não será gerada a partir da solução descartada, mas da última solução aceita.

4.2.2 NA LAMINAÇÃO

Conforme já explicado, a solução a ser modificada pelo simulated annealing da laminação é apenas uma lista de faixa de bitola que define a prioridade de processamento das mesmas. A partir desta lista, o laminador vai sendo preparado para as diferentes faixas, e em cada uma delas, o processamento é feito até que não haja nenhuma ordem dessa faixa disponível para laminação.

Assim, definiu-se, de forma semelhante à fundição, que seriam sorteadas duas faixas de bitola da lista e todo o trecho entre elas seria invertido[1]. Para garantir a geração de uma solução não muito diferente da atual, aqui também é definido um tamanho máximo para a o trecho invertido. Este limite, obviamente, é independente daquele definido para a laminação.

4.3 GERAÇÃO DE SOLUÇÃO SEMENTE

Devido a forma como o simulated annealing foi estruturado, é necessário que seja definido também uma solução semente para a fundição e outra para a laminação.

No caso da fundição, a solução semente é definida para ser utilizada apenas uma vez, no início do programa. Já para a laminação, onde seu simulated annealing é executado a cada mudança de solução para a fundição, a semente será utilizada muitas vezes.

4.3.1 PARA A FUNDIÇÃO

Procurou-se gerar uma boa solução semente para a fundição, de forma que o simulated annealing, começando de uma solução boa, fosse tentando melhorá-la ainda mais, gerando soluções vizinhas não muito distantes uma das outras (vide geração da solução vizinha na fundição).

---

[1] Este tipo de geração de vizinhança é frequentemente encontrado em estudos com o Simulated Annealing (e.g. CHWIF e BARRETO (1994) e PRESS (1994)).

O critério concebido foi a ordenação das fornadas segundo a folga dinâmica total de suas ordens componentes, considerando-se também a quantidade de cada uma. As fornadas com menor folga dinâmica total, calculada multiplicando-se o número de dias de folga de cada ordem pela sua quantidade e obtendo-se um total para a fornada, serão processadas primeiro.

4.3.2 PARA A LAMINAÇÃO

Na laminação, a solução semente será aproveitada do último simulated annealing realizado para a própria laminação. Gerada uma solução vizinha para a fundição, a solução semente para o simulated annealing da laminação será a última solução para laminação aceita. Esta aceitação terá ocorrido no simulated annealing para a laminação com a solução para a fundição anterior.

Uma vez que o método de obtenção de vizinhanças para a fundição gera soluções próximas à solução geradora, uma solução para laminação que 'casou' bem com a última, terá grandes chances de 'casar' bem com a solução vizinha gerada. Assim, tem-se um bom método de geração de solução semente para a laminação.

4.4 TABELA DE RESFRIAMENTO

Na definição da tabela de resfriamento deve-se definir, para os dois simulated annealings de forma independente, o valor inicial do parâmetro T, a função temperatura, T(t), que determina como a temperatura será mudada, o número de iterações, N(t), a ser realizado a cada temperatura, e o critério de parada para encerrar o algoritmo.

Neste problema, a execução de simulated annealings para a laminação com várias soluções para a fundição diferentes permite uma melhor exploração do espaço de soluções do que se fossem feitos simulated annealings para a laminação com poucas soluções para a fundição diferentes, mas com maior procura de uma boa solução na laminação. Isto é, se apenas 40 soluções completas diferentes puderem ser geradas, é preferível que sejam geradas 5 soluções para laminação em cada simulated annealing dela, realizando-se 8 simulated annealings com soluções para fundição diferentes do que gerar-se 8 soluções de laminação para cada uma das 5 soluções para fundição diferentes. Assim é, porque existe uma forte dependência entre a uma solução de laminação e sua

solução de fundição correspondente. Uma mudança na solução da fundição provoca a possibilidade de busca de soluções para laminação em locais diferentes do espaço de soluções. Além disso, o número de fornadas diferentes é bem maior que o número de faixas de bitola.

Assim, os parâmetros que definem o número de soluções vizinhas geradas em cada um dos simulated annealings serão definidos de forma a privilegiar uma maior geração de soluções para a fundição.

4.4.1 PARA O SIMULATED ANNEALING COMPLETO

A temperatura inicial e outros componentes da tabela de resfriamento, serão realmente fixadas durante a execução do programa, em função dos resultados que forem sendo obtidos. Por ora, pode-se fazer apenas definições qualitativas para esses parâmetros.

Uma boa semente para a fundição foi estabelecida. A temperatura inicial, então, não pode permitir que soluções muito piores sejam aceitas livremente, caso contrário, aquela solução não terá muita influência no prosseguimento da busca[2]. Mas deve ainda permitir que sejam aceitos aumentos de custo para que o programa não fique 'preso' num mínimo local. Assim, determinou-se que um aumento de 6% na função custo deve ter, na temperatura inicial, cerca de 35% de probabilidade de se aceito.

A função de temperatura utilizada será proporcional, isto é, $T(t + 1) = \alpha T(t)$ onde α é constante. O valor de α será definido na execução do programa, conforme os resultados que forem sendo obtidos. Vale lembrar que valores típicos para α usados na prática estão entre 0,8 e 0,99 e que o resfriamento lento, isto é, com um α próximo a 1 e com um grande número de passos de temperatura, favorece a convergência para a solução ótima.

O número de soluções geradas para a fundição em cada valor de T, N(t), será proporcional ao tamanho de fornadas e constante em todos os passos de temperatura. Sua fixação, conforme explicado, deverá favorecer a geração de um grande número de soluções para a fundição. Um limitador do número de soluções aceitas em cada passo de

---

[2] Neste caso, a temperatura baixa é especialmente importante, devido ao grande espaço de soluções viáveis.

temperatura será fixado em cerca de 40% de N. Caso este número seja alcançado, deve-se diminuir a temperatura.

A condição de parada da busca será a passagem por um valor de temperatura sem que nenhuma solução seja aceita.

4.4.2 PARA O SIMULATED ANNEALING DA LAMINAÇÃO

Da mesma forma que no simulated annealing completo, a boa solução semente para a laminação torna mais interessante a utilização de uma temperatura inicial não muito elevada. Como no caso anterior, um incremento na função custo de 6%, na temperatura inicial deverá ter cerca de 35% de probabilidade de ser aceito.

A função de temperatura também será proporcional, com um α bem próximo do anterior. O número de passos de temperatura será elevado, favorecendo também o resfriamento lento.

O número de soluções geradas em cada valor de T, N(t), será proporcional ao número de faixas de bitola e constante em todos os passos de temperatura. Seu valor final será mais limitado, em função das razões já discutidas. Existirá também um limitador do número de soluções aceitas em cada passo de temperatura, e será igualmente fixado em cerca de 40% de N.

A condição de parada da busca também será a passagem por um valor de temperatura sem que nenhuma solução seja aceita.

Abaixo resume-se as definições feitas para as duas tabelas de resfriamento.

Tabela 4.1: Definições para a tabela de resfriamento.

	Simulated Annealing Completo	**Simulated Annealing Laminação**
T inicial	35% de probabilidade de aceitação para 6% de aumento no custo	35% de probabilidade de aceitação para 6% de aumento no custo
T(t)	$T(t + 1) = \alpha T(t)$, com α próximo a 1	$T(t + 1) = \alpha T(t)$, com α próximo a 1
número de passos de temperatura	alto	alto
N(t)	alto, proporcional ao número de fornadas	mais baixo, proporcional ao número de faixas de bitola
limite de soluções aceitas	40% de N em cada temperatura	40% de N em cada temperatura
condição de parada	temperatura sem nenhuma solução aceita	temperatura sem nenhuma solução aceita

5.

IMPLEMENTAÇÃO DO PROGRAMA

5.1 INTRODUÇÃO

Neste capítulo são feitas, inicialmente, considerações gerais sobre a implementação do Sequenciador, tais como o equipamento utilizado, a escolha da linguagem e as características gerais do programa. Em seguida é descrita a forma de leitura dos dados utilizados no processamento.

Em sua parte mais extensa são descritos os algoritmos do programa. Por fim trata dos testes e da depuração, realizados antes da sua utilização.

5.2 EQUIPAMENTO E PROGRAMAS UTILIZADOS

O equipamento utilizado, tanto para o desenvolvimento do Sequenciador quanto para sua utilização, foi um micro computador 486 DX4 100 MHz, com 16 Mb de memória RAM. O sistema operacional utilizado foi o Microsoft Windows 95.

Para a implementação foi escolhida a linguagem de programação C. Tal escolha deu-se em função da grande velocidade de execução oferecida pela linguagem, que poderia proporcionar a possibilidade de geração de um grande número de soluções num um tempo de processamento reduzido, contribuindo fortemente para o bom desempenho do programa.

Além disso, a linguagem permite grande flexibilidade, tanto no acesso e gravação de dados em disco quanto na manipulação de endereços e alocação dinâmica de memória, além de ser uma linguagem estruturada.

Como desvantagens, o desenvolvimento do programa é difícil e lento, bem como sua depuração.

Devido à maior facilidade e rapidez em se trabalhar com todos os dados de entrada na memória em relação a mantê-los parcialmente em disco, optou-se por codificar o programa daquela forma. Supôs-se que, uma vez que o micro possuía 16 Mb de memória RAM, não haveria problema de espaço em memória.

Mas, na verdade, ao se escrever um programa em linguagem C, padrão ANSI[1], a memória disponível para a execução do mesmo é apenas 64 Kb.

---

[1] Um programa em linguagem C padrão ANSI utiliza apenas comandos reconhecidos por qualquer compilador C. Escrito dessa forma, também pode ser compilado por compiladores C++.

O problema foi resolvido utilizando-se um compilador obtido na Internet, o DJGPP, que permite a utilização de toda a memória disponível, fazendo inclusive uma simulação de memória no próprio disco rígido, caso esta ainda não seja suficiente.

Assim, para o desenvolvimento, testes e depuração do programa foi utilizado o *software* Borland C++. Tais teste foram realizados com apenas uma parte das ordens que deveriam ser processadas, tornado possível a execução com este compilador. As vantagens de sua utilização são a ótima interface permitida e as facilidades de depuração que o sistema oferece.

Para a execução com todas as ordens de produção desejadas, utilizou-se o DJGPP, executado a partir do *prompt* MS-DOS, através de linha de comando, mas permitindo a execução adequada do programa.

5.3 ENTRADA DOS DADOS

Os dados obtidos na Lamina foram enviados em planilha Excel e continham um conjunto de ordens de produção e os tempos de produção dos equipamentos.

A planilha de ordens de produção continha como campos o número da ordem e a parcela desta, seu grupoliga, a faixa de bitola, a data de entrega, a quantidade pedida e o acabamento desejado. A partir dela, foi criado um arquivo em formato texto, que seria lido pelo Sequenciador.

Já que o conteúdo desse arquivo ficaria todo na memória, decidiu-se fazer uma série de codificações para que os dados ocupassem menos espaço. Assim, o arquivo texto, chamado ORDENS, não contém os campos número da ordem e parcela. Tais campos apenas identificam as ordens, não sendo necessários na execução do programa. A identificação seria realizada através da própria ordem do arquivo, que continuou a mesma da planilha.

O campo 'grupoliga', que na planilha tinha os quatro primeiros dígitos identificando o grupo e outros nove identificando a liga, foi substituído no arquivo por dois campos separados. O primeiro campo identifica o grupo através de um número inteiro, a partir de zero, isto é, o primeiro grupo que aparece na planilha foi substituído por zero, o segundo por um, e assim sucessivamente. O segundo campo identifica a liga e foi substituído por um caracter. Codificações semelhantes foram feitas para a faixa de

bitola e acabamento. Com essa codificação, o acesso aos tempos de produção poderia ser feito muito facilmente, pois o próprio valor do campo seria utilizado como índice dos vetores de tempos de produção.

A data de entrega da planilha foi substituída por um prazo de entrega, em horas, calculado a partir de uma data fixa. A quantidade foi passada de quilos para dez quilos.

Os tempos de produção, obtidos dos arquivos TP_GRUPO e TP_FAIXA, foram todos passados para horas. A produtividade do laminador, por exemplo, foi passada de toneladas por hora para horas por dez quilos.

Os dados sobre custos e limites do forno, obtidos no arquivo OUTROS, estão em unidades compatíveis com as demais utilizadas.

5.4 ROTINAS DESENVOLVIDAS

O programa foi desenvolvido de forma estruturada, com rotinas separadas responsáveis pela execução de cada etapa do programa. Os nomes de campos e de funções são bastante extensos e indicam sua finalidade, facilitando o entendimento. Nele existe um grande número de comentários, tanto nas definições de variáveis, explicando suas características, como no início e ao longo da funções, descrevendo seu funcionamento.

Utilizou-se também o deslocamento das linhas de código, de forma a facilitar o entendimento de comandos como if, for, do while e outros. A diretiva #define também foi utilizada. Todas essas característica facilitam tanto o entendimento quanto a alteração do código fonte. Ao todo, o Sequenciador possui cerca de 1550 linhas.

No acesso aos vetores, foi utilizada a referência a ponteiro, do tipo '*(vetor_ex + i)', ao invés da referência a vetor, do tipo 'vetor_ex[i]', já que, segundo SCHILDT (1991), a primeira proporciona maior velocidade de execução.

Apresentam-se agora, para as principais rotinas do programa, seus algoritmos em pseudocódigo.

Rotina principal

A rotina principal do programa, função Main em C, executa a chamada das rotinas que realizam as várias etapas do trabalho. Nela pode-se ver a estrutura de duplo simulated annealing.

Tabela 5.1: Rotina principal.

```
Lê os dados sobre ordens, tempos de produção e custos;
Gera solução semente da fundição;
Inicializa temperatura da fundição;

Para cada passo de temperatura
  Para cada uma das Nfund vezes em cada temperatura

    Calcula instante final de desbaste de cada ordem;
    Apenas na primeira vez
      Calcula horizonte ideal de processamento;
    Gera solução semente da laminação;
    Avalia solução;
    Salva solução como melhor do s.a. da laminação;
    Inicializa temperatura da laminação;

    Para cada passo de temperatura
      Para cada uma das Nlam vezes em cada temperatura

        Gera solução vizinha na laminação;
        Avalia solução;
        Compara com melhor do s.a. da laminação;
        Se aceitar
          Salva solução como melhor do s.a. da laminação;
        Se ultrapassou limite de soluções aceitas
                                        por passo de temperatura
          Diminui temperatura;

      Se nenhuma solução foi aceita
        Final de s.a. da laminação;

    Compara melhor solução do s.a. da laminação com
                              melhor solução do s.a. da completo
    Se aceitar
      Salva solução como melhor do s.a. completo;
    Gera solução vizinha na fundição;
    Se ultrapassou limite de soluções aceitas
                                        por passo de temperatura
      Diminui temperatura;

  Se nenhuma solução foi aceita
    Final de s.a. completo;
```

Leitura dos dados sobre ordens, tempos de produção e custos

Na leitura das ordens, os dados são obtidos do arquivo ORDENS e passados para o vetor de estruturas dados_ord ordens. A estrutura dados_ord é composta pelos campos necessários para cada ordem: grupo, liga, faixa de bitola, acabamento, quantidade e prazo. Para o funcionamento adequado do programa, o arquivo deve estar com as ordens de mesma liga agrupadas, isto é, em sequência no arquivo.

Os dados sobre tempos de produção dos equipamentos também são passados para vetores. Os dados sobre tempo de fundição e desbaste de cada grupo, por exemplo, são lidos para um vetor de estruturas grupo, a qual contém o tempo de fundição, desbaste e seis tempos de acabamento. Assim, cada linha deste vetor representa os tempos de produção de um grupo e o acesso a esses tempos pode ser feito com os próprios valores dos campos grupo e acabamento no vetor ordens.

O mesmo ocorre para os tempos de produção e preparação para cada faixa de bitola.

Tabela 5.2: Leitura dos dados sobre ordens, tempos de produção e custo.

Abre arquivo ORDENS e lê dados para vetor ordens;
Abre arquivo TP_GRUPO e lê dados para vetor tp_grupo;
Abre arquivo TP_FAIXA e lê dados para vetor tp_faixa;
Abre arquivo OUTROS e lê dados para variáveis;

Geração da solução semente da fundição

No início da geração da solução semente fundição, é criado um vetor de ponteiros para a estrutura dados_ord, onde cada elemento aponta para uma ordem no vetor ordens. Este vetor, chamado ordens_liga_folga, que já herdará o agrupamento por liga do vetor ordens, é então ordenado, em cada liga, por folga dinâmica. Esta e outras ordenações realizadas no programa são feitas através do algoritmo de ordenação Quicksort.

Em seguida são definidas as fornadas, de acordo com a regra estabelecida no capítulo 3. Isto é feito através da criação do vetor de estruturas fornada[2] sol_fund, no

[2] A estrutura fornada possui dois ponteiros para ponteiro para a estrutura dados_ord, que apontam para o início e o final de cada fornada.

qual cada linha aponta o início e o final de cada fornada no vetor ordens_liga_folga, conforme ilustra o desenho a seguir.

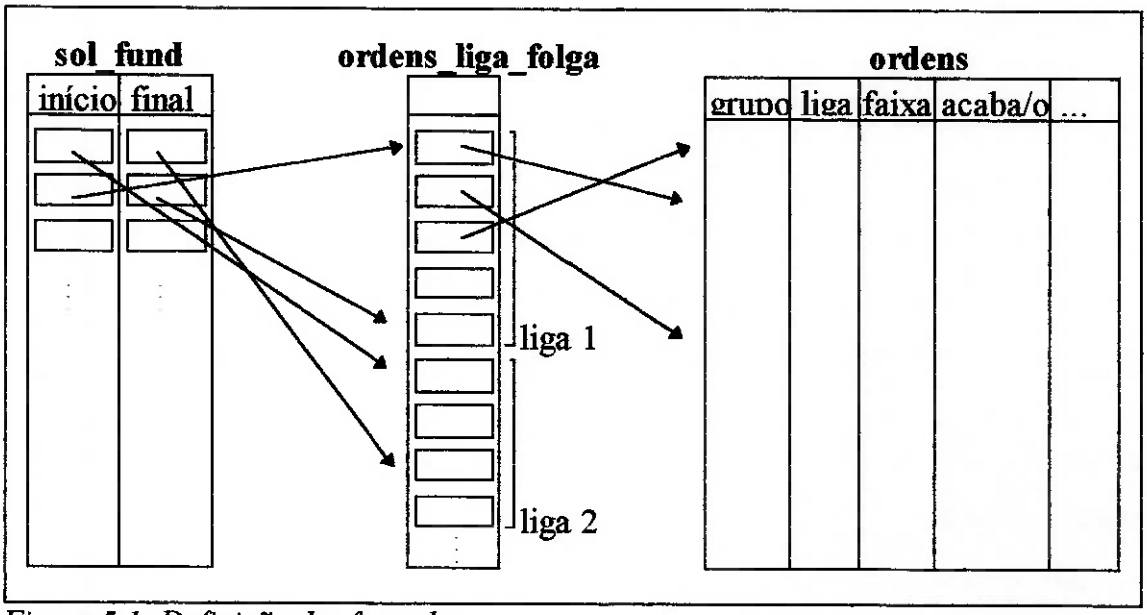

Figura 5.1: Definição das fornadas.

Por fim, este vetor é ordenado por folga total, fixando-se a ordem de processamento semente das fornadas.

Tabela 5.3: Geração da solução semente da fundição.

Cria vetor de ponteiros para dados_ord ordens_liga_folga;
Ordena vetor ordens_liga_folga por folga dinâmica dentro de cada liga;
Define fornadas através do vetor sol_fund, que aponta início e final de fornada em ordens_liga_folga;
Ordena fornadas, isto é, sol_fund por folga total;

Calcula instante final de desbaste

Para cada solução para fundição criada, é necessário, para cada ordem, calcular o instante de término da etapa de desbaste. Tal instante será utilizado tanto na definição dos lotes de laminação quanto na avaliação da solução, isto é, no cálculo do custo associado a solução.

Este instante e o instante de término da laminação são calculados em relação ao início da fundição da primeira fornada, que é o instante zero da simulação. A estrutura

dados_ord, além dos campos com informações sobre liga, faixa de bitola, quantidade, etc., possui os campos fim_desb e fim_lam, os quais serão preenchidos com os instantes de término do desbaste e da laminação.

Tabela 5.4: Calcula instante final de desbaste.

```
Para cada fornada
   Calcula o instante de término da fornada;
   Para cada ordem da fornada
      Acrescenta o tempo de desbaste e coloca em fim_desb;
```

Geração da solução semente da laminação

Antes da geração da solução semente da laminação, é criado outro vetor de ponteiros para dados_ord, o ordens_faixa_fim_desb, semelhante a ordens_liga_folga, mas que desta vez é agrupado por faixa de bitola e ordenado por instante final de desbaste dentro de cada faixa. Este vetor será utilizado na definição dos lotes de laminação.

Em seguida, é criado o vetor faixas_seq_proc, que nada mais é que a lista que define a sequência de faixas de bitola pela qual o laminador deve ser preparado. Trata-se de um vetor de ponteiros para ponteiro para dados_ord, onde cada elemento aponta para o primeiro elemento de cada faixa de bitola em ordens_faixa_fim_desb. Este vetor é obtido da melhor sequência encontrada no simulated annealing da laminação anterior.

Por fim é chamada a rotina para o cálculo do instante final de laminação.

Tabela 5.5: Geração da solução semente da laminação.

```
Ordena vetor ordens_faixa_fim_desb por folga dinâmica dentro
                                                de      cada
faixa;
Cria vetor faixas_seq_proc, a partir da melhor sequência do
                                        s.a. da laminação anterior;
Calcula instante final de laminação de cada ordem;
```

Cálculo do instante final de laminação

Esta rotina realiza, separadamente, a composição dos lotes de laminação e a definição da sequência de processamento das ordens em cada lote, de acordo com a lista

de prioridades de faixas de bitola e com a regra heurística estabelecida. Tal separação é possível, já que a ocorrência de fila zero, condição de mudança de lote de laminação, não depende da sequência de processamento das ordens num lote.

Utilizando o vetor ordens_faixa_fim_desb e o vetor que define a prioridade das faixas de bitola, a rotina vai preenchendo o vetor sol_lam, o qual aponta para ordens_faixa_fim_desb indicando o início e o final de cada lote de laminação. A rotina leva em consideração que a fila de uma faixa de bitola pode estar vazia no momento em que for sua vez de ser preparada[3], e passa automaticamente para a próxima faixa da lista sem computar uma preparação. Caso não haja nenhuma ordem em nenhuma fila ao final de um lote, o laminador é preparado para a faixa de bitola da ordem que for sair primeiro do desbaste.

Na definição da sequência de processamento em cada lote, a rotina considera, obviamente, a possibilidade de uma ordem ainda não ter saído do desbaste ao chegar a sua vez de ser processada. Isto é, o critério de processamento é menor folga dinâmica dentre as ordens que estão na fila, e não das ordens que estão no lote. Esta parte da rotina também calcula o instante final de laminação. Para realizar tais tarefas, ela cria um vetor de ponteiros para a estrutura dados_ord, o ordens_lote_folga, copiando-o de ordens_faixa_fim_desb e ordenando-o, em cada lote, por ordem de folga dinâmica. Em seguida, um outro vetor do mesmo tipo, o ordens_seq_proc_lam, vai sendo gerado, fixando a ordem de processamento das ordens. Este último vetor só define a ordem de processamento das ordens dentro de um lote. A ordem de processamento dos lotes é definida pelo vetor sol_lam.

[3] Na verdade, a rotina verifica se chegará alguma ordem na fila até o final da preperação.

Tabela 5.6: Cálculo do instante final de laminação.

```
Cria vetor que indicará a ordem de cada faixa de bitola
                                que está sendo
processada;
Seleciona primeira ordem a entrar num lote (considera o
                                parâmetro espera);
Para cada lote
   Coloca primeira ordem do lote em sol_lam na solução;
   Para todas as ordens deste lote
      Coloca ordem em sol_lam;
      Pega próxima ordem;

Cria ordens_lote_folga a partir de ordens_faixa_fim_desb;
Ordena ordens_lote_folga por folga dinâmica
                                dentro de cada lote;
Para cada lote
   Para cada ordem do lote
      Se já estiver disponível
         Coloca ordem em ordens_seq_proc_lam;
```

Geração da solução vizinha na laminação

A geração da solução vizinha, tanto na laminação quanto na fundição, é sempre feita a partir da última solução aceita, e não de uma solução que tenha acabado de ser desprezada. Assim, inicialmente, o vetor contendo a lista de prioridades de faixas de bitola correspondente à última solução aceita do simulated annealing da laminação é copiado. Em seguida, são sorteadas duas faixas da lista e a prioridade neste trecho é invertida, conforme explicado no capítulo anterior. Este trecho não pode ter um tamanho superior a três faixas. Por fim, a rotina de cálculo do instante final de laminação é chamada.

Tabela 5.7: Cálculo do instante final de laminação.

```
Copia melhor solução do simulated annealing da laminação
                                até o momento;
Sorteia duas faixas na lista;
Inverte prioridades do trecho definido;
Calcula instante final de laminação de cada ordem;
```

Geração da solução vizinha na fundição

Muito semelhante à rotina anterior, a solução vizinha na fundição também é obtida a partir da melhor solução para fundição encontrada até o momento. O trecho de fornadas cuja ordem de processamento será invertida, porém, pode ser de até cinco fornadas.

Tabela 5.8: Geração da solução vizinha na fundição.

Copia melhor solução do simulated annealing da laminação até o momento;
Sorteia duas faixas na lista;
Inverte prioridades do trecho definido;

Avaliação da solução

O cálculo da função custo é feito principalmente com base nos tempos calculados, indicando os instantes finais de desbaste e laminação. Com esses tempos, podem ser calculados os tempos de atraso, adiantamento e de estoque intermediário para cada ordem, além do horizonte de processamento. O número de preparações na laminação também é obtido, podendo-se assim, avaliar o desempenho de uma solução em cada um desses critérios e obter um total geral através das taxas monetárias para cada um deles, obtidas do arquivo OUTROS no início do programa.

Tabela 5.9: Avaliação da solução.

Avalia desempenho em atendimento aos prazos;
Avalia desempenho em estoque intermediário;
Avalia desempenho em horizonte de processamento;
Avalia desempenho em número de preparações na laminação;

Salvamento da solução

Na rotina principal do programa é feito o salvamento, em memória, da última solução aceita, tanto no simulated annealing da laminação quanto no simulated annealing completo.

No simulated annealing da laminação, são salvos o vetor de prioridades das faixas de bitola, faixas_seq_proc, o vetor que define os lotes de laminação e sua ordem de

processamento, sol_lam, o vetor que define a ordem de processamento das ordens em cada lote, ordens_faixa_seq_proc_lam e, é claro, o valor da função custo obtido para tal solução. No simulated annealing completo são salvos, além dos vetores citados, que correspondem à melhor solução encontrada no simulated annealing da laminação, o vetor sol_fund, que define as fornadas e ordem de processamento das mesmas.

Geração de números aleatórios

A geração de números aleatórios, tanto para o sorteio de fornadas ou faixas de bitola para geração de soluções vizinhas quanto para o sorteio para decisão de se aceitar uma solução pior, é realizada através da função ran3, obtida em Numerical Recipes in C. Esta função, cada vez que é chamada, devolve um número em formato ponto flutuante entre 0,0 e 1,0, com distribuição uniforme. Para inicialização da sequência, deve-se passar um número negativo como argumento.

Rotinas de ordenação

Todas as rotinas de ordenação do programa utilizam o algoritmo Quicksort, eficiente método de ordenação por particionamento baseado no princípio da permutação. Neste algoritmo, pares de elementos do vetor vão sendo trocados e o vetor subdividido, sendo o método aplicado novamente em cada uma das partes. O resultado é um ganho surpreendente de velocidade em relação a algoritmos baseados nos princípios de seleção e inserção, como o Bublesort, fazendo com que esse seja o melhor método de ordenação de vetores até agora conhecido.

Saída dos dados

Os dados de saída gerados pelo programa serão diferentes nas diversas fases de execução do programa, isto é, os dados gravados ou mostrados em tela durante as execuções para teste serão diferentes daqueles gravados ou mostrados no acerto da tabela de resfriamento ou na execução final com a tabela acertada. De uma forma geral, os dados serão gravados em arquivos textos, de forma a poderem ser lidos por programas como Excel ou Access, e poderem ser compilados e apresentados graficamente.

5.5 TESTES REALIZADOS

Após a codificação, foram realizados os testes de funcionamento do Sequenciador. Foram realizados com o programa Borland C++ for Windows, assim como nos ensaios necessários para dirimir dúvidas sobre o funcionamento da linguagem, quando isto não era possível através da bibliografia disponível.

Nos testes finais foram utilizadas parte das ordens de produção obtidas na Lamina, já devidamente formatadas no modo requisitado pelo Sequenciador. As 52 ordens geraram, na solução semente, 6 fornadas e 5 lotes de laminação. Um conjunto de valores baixos foram utilizados na tabela de resfriamento. Os tempos de produção fornecidos pela empresa também entraram nesses testes.

Utilizando-se o Microsoft Excel, foram definidas as fornadas e os lotes de laminação e calculados os instantes finais de desbaste e laminação para a solução semente gerada, do modo que deveria ser feito pelo programa. Definiu-se as taxas monetárias para avaliação do desempenho e calculou-se também a função custo.

As ferramentas de depuração do Borland C++ permitiram a execução do programa passo a passo, visualização das variáveis de memória e cadeia de funções que foram sendo chamadas, além do estabelecimento de pontos de pausa na execução. Com o uso delas e a comparação dos resultados fornecidos com os calculados no Excel, foram sendo retirados todos os erros do programa.

Alguns erros foram difíceis de se descobrir. Em uma função de ordenação, por exemplo, o cálculo da folga dinâmica a partir dos tempos de produção e prazos era feito e atribuído a uma variável tipo *float*. Ao se comparar o valor de tal variável com a expressão com a qual o mesmo havia sido calculado, o programa entendia como diferentes. Os dados apresentados pelo depurador, porém, eram visualmente iguais. Após entendido o que estava acontecendo, atribuiu-se também a expressão a uma variável, antes da comparação, resolvendo-se o problema.

Também foram conferidas as rotinas de geração soluções vizinhas e salvamento de soluções aceitas. Por fim, utilizou-se o compilador DJGPP para execução do programa com todas as ordens desejadas[4]. O compilador, porém, não aceitava a função exp(), usada no cálculo de exponencial para determinação da probabilidade de aceitação

[4] A execução com muitas ordens no Borland C++, mesmo com a opção de maior utilização de memória, causava travamento do micro.

de aumento no custo. A função pertence a biblioteca padrão de funções matemáticas da linguagem C e havia funcionado perfeitamente no Borland C++. Foi necessário, então, a codificação de uma função que calculasse o exponencial de um número real, através da série de Taylor.

Com programa compilando adequadamente, as execuções puderam ser realizadas, após a eliminação de um último erro de lógica, ainda encontrado neste ponto do trabalho.

Nos anexos, encontra-se o código fonte do programa, com a tabela de resfriamento e a gravação dos dados em disco utilizadas em uma das execuções.

6.

SIMULAÇÃO

6.1 INTRODUÇÃO

Este capítulo apresenta os dados utilizados na simulação e os resultados de cada uma das execuções realizadas com diferentes tabelas de resfrimento, através da utilização de gráficos de linhas.

6.2 DADOS UTILIZADOS

6.2.1 ORDENS DE PRODUÇÃO

Encontra-se nos anexos a lista de ordens de produção utilizadas na simulação, tal qual enviada pela empresa. Contém 682 ordens, num total de 341 toneladas de aço, estando todas suas datas de entrega compreendidas num período de 8 semanas. São 34 ligas diferentes, pertencentes a 5 grupos e a 8 faixas de bitola. Encontram-se nelas, também, todas as seis opções de acabamento.

Conforme já dito, os dados foram passados deste para um formato mais compacto, de forma a ocupar pouco espaço em memória, gerando o arquivo ORDENS, a ser lido pelo Sequenciador. O trecho inicial deste arquivo também encontra-se nos anexos. Nele pode-se observar que, apesar de não conter os números de cada ordem de produção, é possível se identificar o mesmo através da ordenação do arquivo. Ambos estão exatamente na mesma ordem, e, conforme exigido pelo Sequenciador, com as ordens de mesma liga agrupadas.

6.2.2 TEMPOS DE PRODUÇÃO

A tabela abaixo, obtida na Lamina, mostra os tempos de fundição, desbaste e acabamento de cada grupo de ligas. Em seguida, a tabela 6.2 mostra os tempos de laminação e preparação do laminador para cada faixa de bitola. Os dados dos arquivos lidos pelo Sequenciador, TP_GRUPO e TP_FAIXA[1], encontram-se nos anexos. Neles não são necessários o código do grupo e da faixa de bitola. Os campos do arquivo lido ORDENS que contém os dois dados, já indicam diretamente a linha do vetor, para onde o arquivo é lido, de onde os dados devem ser obtidos.

---

[1] Nesses arquivos alguns dados estão em unidades diferentes das daqui apresentadas.

Tabela 6.1: Tempos de fundição, desbaste e acabamento.

Grupo	Tempo de fundição (hs)	Produtividade no desbaste (t/h)	Lead time acabamento (h)					
			bruto	trefilado	descascado	retificado	torneado	fresado
101	8	9,0	40	72	48	64	56	56
103	8	9,0	40	56	48	56	56	56
201	9	10,0	32	56	56	48	56	48
202	9	10,5	24	56	40	48	56	56
3xx	8	9,8	16	72	32	48	-	-

Tabela 6.2: Tempos de preparação e laminação.

Faixa de bitola	Tempo de preparação (h)	Produtividade na laminação (t/h)
01	15	1,65
02	4	1,98
03	9	2,31
04	10	2,19
05	7	1,53
06	8	1,95
07	5	1,47
08	5	1,95

6.2.3 OUTROS

Abaixo estão os dados contidos no arquivo OUTROS, sobre taxas monetárias, ou custos, de atraso e adiantamento, estoque, preparação e horizonte de processamento, além da capacidade e do limite mínimo de ocupação do forno em cada fornada. As taxas foram estimadas para a simulação, estando compatíveis com as definições de solução, geração de solução semente e busca de vizinhança estabelecidas.

Tabela 6.3: Taxas monetárias e limites de ocupação.

capacidade do forno	25 t
ocupação mínima	10 t
taxa de atraso	10 R$/t.h
adiantamento	1 R$/t.h
estoque	1 R$/t.h
preparação do laminador	100 R$
horizonte de processamento	1 R$/h

6.3 RESULTADOS OBTIDOS

Numa execução prévia, percebeu-se que o valor da função custo obtido para uma solução razoável para o processamento das 682 ordens seria de R$ 500.000,00. Pode-se assim, fixar a temperatura inicial para os dois simulated annealings em R$ 30.000,00, de acordo com as definições do capítulo 4. Fixando-se o número de passos de temperatura em 25, obteve-se uma curva de resfriamento adequada com o parâmetro α igual a 0,92.

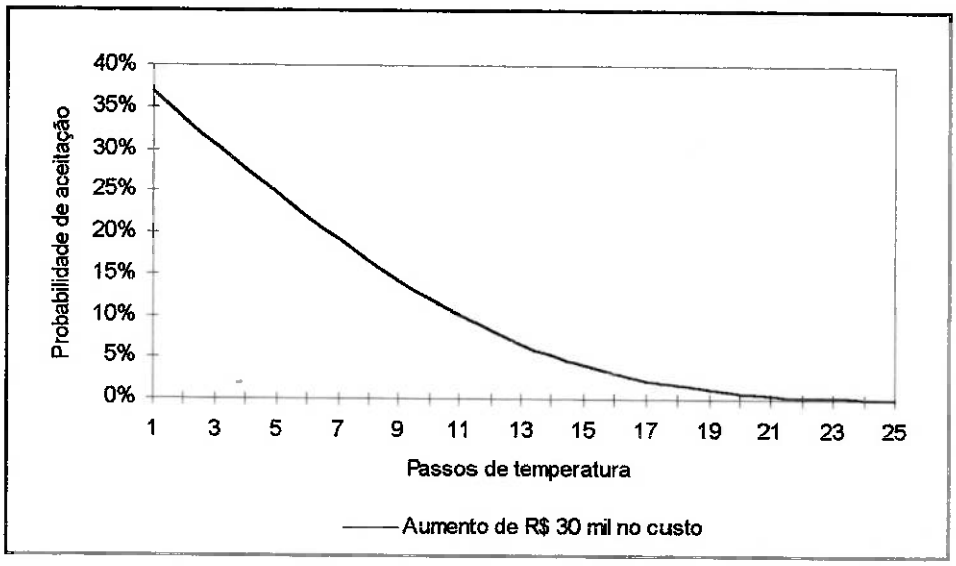

Figura 6.1: Curva de resfriamento para α igual a 0,92.

6.3.1 PRIMEIRA EXECUÇÃO

Fixou-se, então, a tabela de resfriamento para a primeira execução.

Tabela 6.4: Tabela de resfriamento da primeira execução.

	Simulated Annealing Completo	**Simulated Annealing Laminação**
T inicial	R$ 30.000,00	R$ 30.000,00
T(t)	T(t + 1) = 0,92.T(t)	T(t + 1) = 0,92.T(t)
número de passos de temperatura	25	25
N(t)	10	8
limite de soluções aceitas	4 em cada temperatura	3 em cada temperatura
condição de parada	1 passo de temperatura sem nenhuma solução aceita	1 passo de temperatura sem nenhuma solução aceita

Com essa tabela, a execução levou apenas dez minutos. Os valores da função custo para as soluções aceitas no simulated annealing completo aparecem no gráfico a seguir. Nele também aparece o custo da solução semente do simulated annealing da laminação para cada uma das soluções aceitas, além da temperatura do simulated annealing completo.

Os resultados obtidos foram bastante interessantes. A primeira solução completa gerada teve um custo de R$ 590.044,00. O primeiro simulated annealing da laminação encontrou uma solução com custo de R$ 508.418,00. A geração de outras soluções para a fundição e a realização de novos simulated annealing na laminação levou o custo a R$ 497.700,00. Ao todo foram geradas 18.300 soluções completas.

Observando-se o gráfico, percebeu-se a necessidade de uma alteração no programa. Além do salvamento da última solução aceita, nos dois simulated annealings, deveria ser salva também a solução de menor custo encontrada. Tal correção definitiva não foi realizada, mas foi feita uma alteração mais simples e improvisada que permitiu, da mesma forma, o acesso à melhor solução encontrada.

Como o tempo de execução foi muito pequeno, foi realizada uma nova execução com um número maior de soluções geradas.

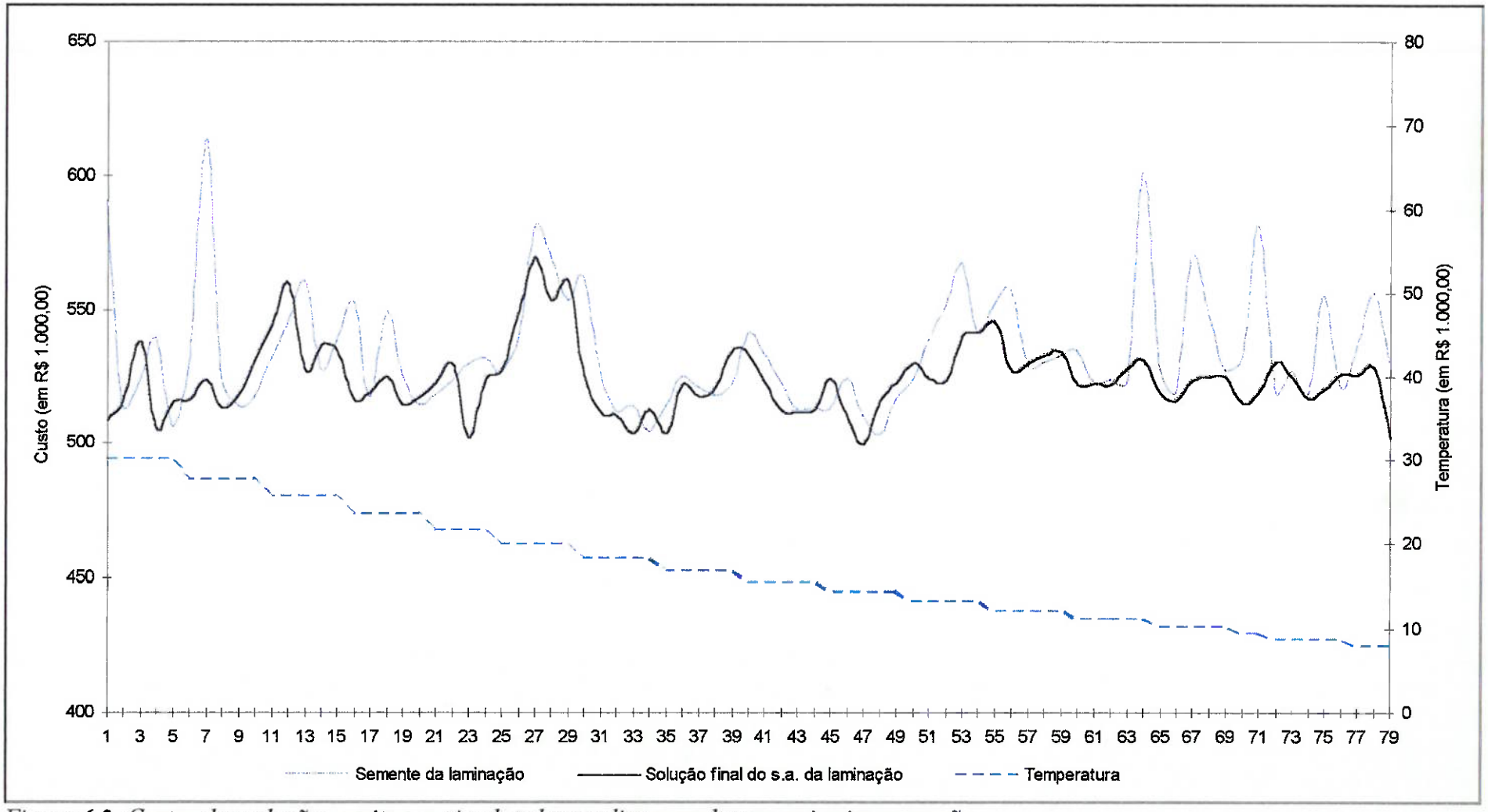

Figura 6.2: Custos das soluções aceitas no simulated annealing completo na primeira execução.

6.3.2 SEGUNDA EXECUÇÃO

Nesta execução, a curva de resfriamento apresentada anteriormente foi mantida, em relação ao número de passos de temperatura e à constante α. A temperatura inicial, porém, foi aumentada para se melhorar a possibilidade de saída de ótimo local.

Aumentou-se também o número de iterações em cada nível de temperatura nos dois simulated annealings. Consequentemente, aumentou-se também o limite de soluções aceitas.

Tabela 6.5: Tabela de resfriamento da segunda execução.

	Simulated Annealing Completo	**Simulated Annealing Laminação**
T inicial	R$ 40.000,00	R$ 30.000,00
T(t)	T(t + 1) = 0,92.T(t)	T(t + 1) = 0,92.T(t)
número de passos de temperatura	25	25
N(t)	20	16
limite de soluções aceitas	6 em cada temperatura	3 em cada temperatura
condição de parada	1 passo de temperatura sem nenhuma solução aceita	1 passo de temperatura sem nenhuma solução aceita

Com essa nova tabela, a execução levou 18 minutos. Desta vez, foram geradas um total de 41.763 soluções. Os resultados encontram-se no gráfico a seguir, de forma idêntica à primeira execução.

O gráfico, dividido em duas partes, apresenta, como era esperado, uma maior instabilidade no custo das soluções nos primeiros passos de temperatura. Na quadragésima oitava solução aceita foi obtida a melhor solução. O simulated annealing da laminação que chegou a esta solução é mostrado no gráfico 6.4.

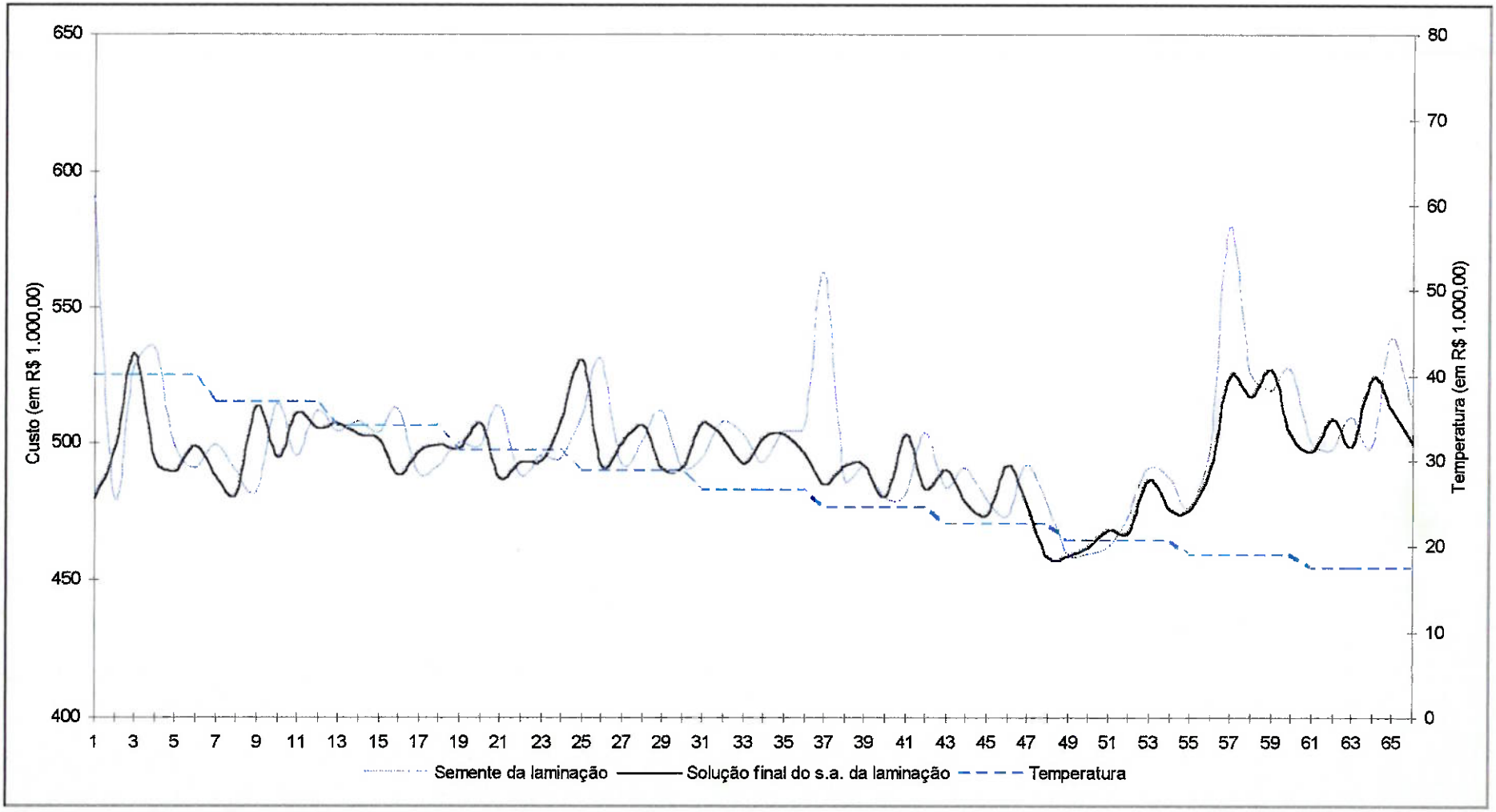

Figura 6.3a: Custos das soluções aceitas no simulated annealing completo na segunda execução.

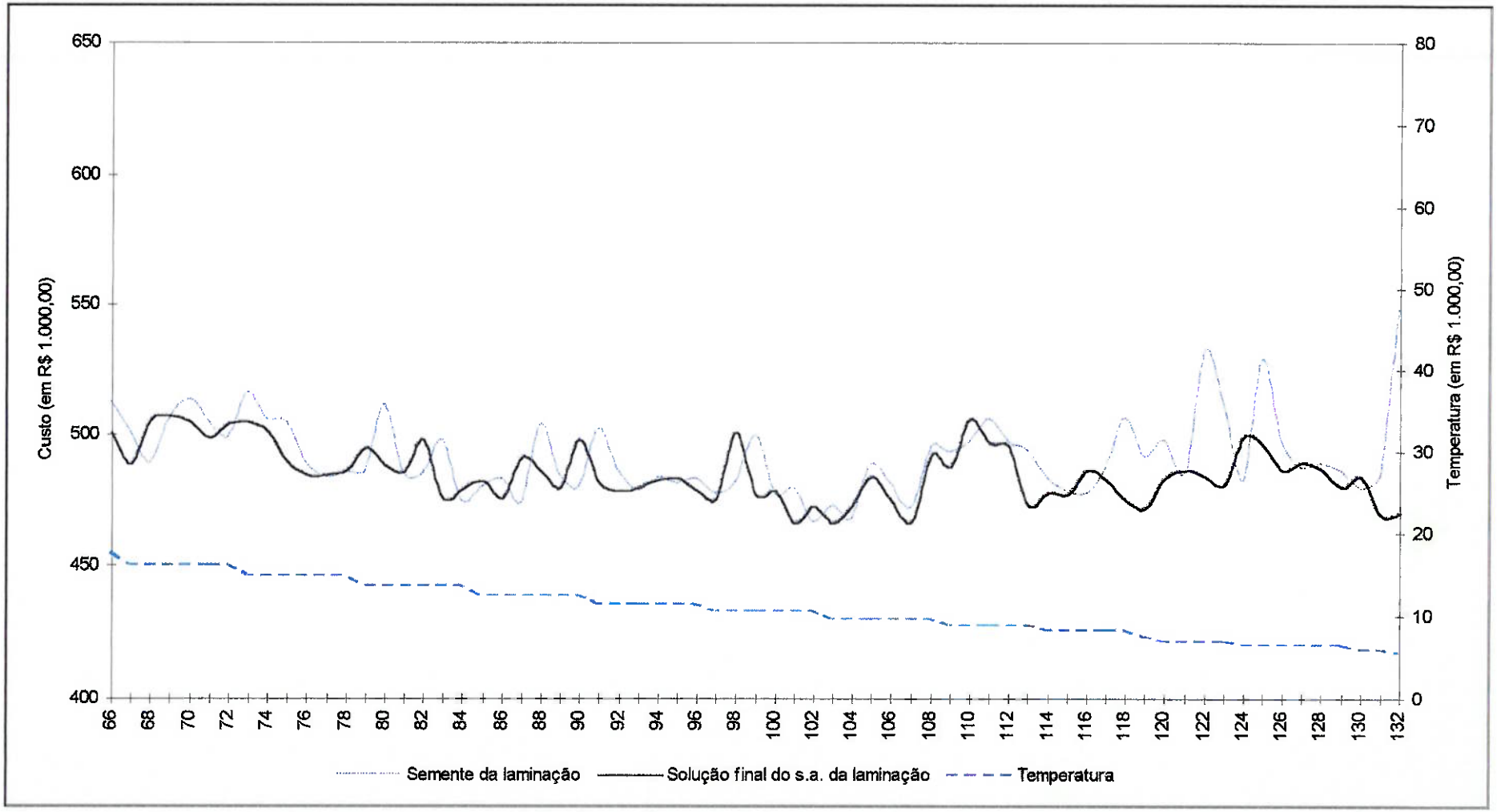

Figura 6.3b: Custos das soluções aceitas no simulated annealing completo na segunda execução.

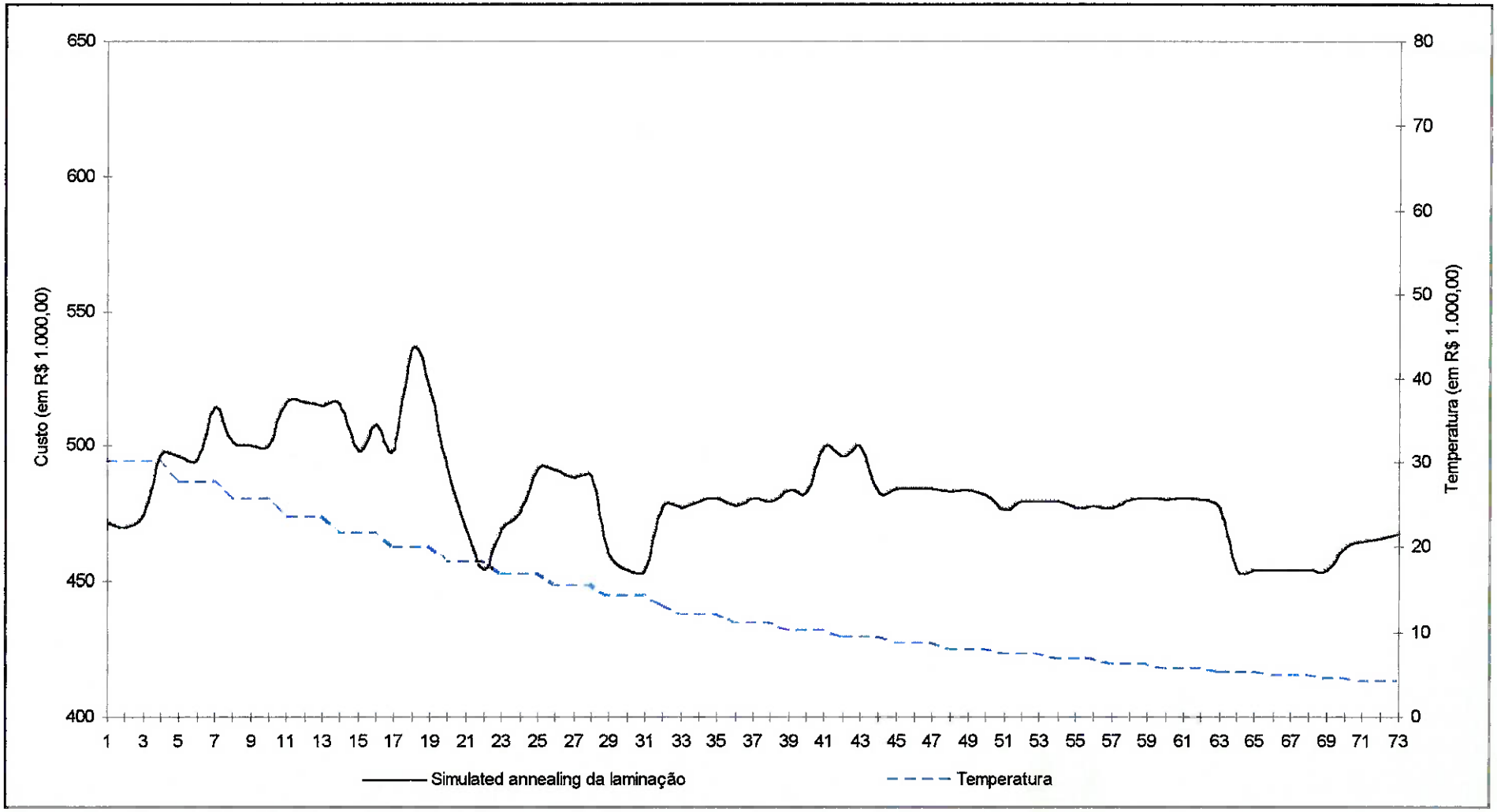

Figura 6.4: Custos das soluções aceitas no melhor s.a. da laminação na segunda execução.

Neste simulated annealing, foi encontrada a solução de menor custo, no valor de R$ 453.770,00.

Como o tempo de execução ainda era pequeno, tentou-se uma nova execução com outra tabela de resfrimento.

6.2.4 TECEIRA EXECUÇÃO

Na terceira execução foi aumentada a temperatura inicial do simulated annealing da laminação.

Para maior geração de soluções, aumentou-se o número de soluções geradas em cada passo de temperatura em ambos os simulated annealings. Aumentou-se também o número de passos de temperatura, já que a probabilidade de aceitação de pelo menos uma solução em cada passo também cresceria. O limite de soluções aceitas em cada passo foi aumentado, igualmente para acompanhar a mudança em N(t).

Tabela 6.7: Tabela de resfriamento da terceira execução.

	Simulated Annealing Completo	**Simulated Annealing Laminação**
T inicial	R$ 40.000,00	R$ 40.000,00
T(t)	T(t + 1) = 0,92.T(t)	T(t + 1) = 0,92.T(t)
número de passos de temperatura	30	30
N(t)	25	20
limite de soluções aceitas	7 em cada temperatura	6 em cada temperatura
condição de parada	1 passos de temperatura sem nenhuma solução aceita	1 passos de temperatura sem nenhuma solução aceita

Em uma hora e dez minutos de execução o programa gerou 164.500 soluções. A melhor solução entre elas, porém, foi superior a R$ 479.000,00. A melhor solução geral encontrada, destarte, ficou sendo a melhor solução da segunda execução, a qual é detalhada nos anexos.

7.

CONCLUSÕES

7.1 CONCLUSÕES E PROPOSTA PARA CONTINUIDADE

Os resultados obtidos foram bastante satisfatórios. Os custos unitários utilizados penalizavam mais fortemente os atrasos na entrega. Coerentemente, a solução semente da fundição privilegiava o bom desempenho neste critério, ordenando as fornadas por folga dinâmica total. Mesmo assim, o Sequenciador reduziu o custo da solução semente de R$ 590.044,00 para R$ 453.770,00, representando uma ganho de 23%.

Vale lembrar que, na primeira execução, o primeiro simulated annealing da laminação levou o custo até R$ 508.418,00. Pode-se atribuir essa facilidade de redução ao fato de a solução semente inicial da laminação não ser muito boa. Ainda, assim, a redução obtida a partir deste valor é de mais de 10%.

Em relação ao número de soluções geradas e ao tempo de execução, os resultados foram ainda melhores.

Normalmente, o cálculo do custo da solução vizinha obtida em implementações do simulated annealing é feito já com base no custo da solução geradora, avaliando-se a modificação realizada. Numa aplicação do simulated annealing ao problema do caixeiro viajante, apresentada em PRESS (1994), uma solução vizinha é gerada invertendo-se um trecho do caminho atual e a função custo, que no caso é a distância percorrida, é facilmente calculada avaliando-se a mudança ocorrida no caminho entre apenas dois pares de cidades.

Neste problema, com as definições de solução e obtenção de vizinhança estabelecidas, não é possivel realizar a avaliação da solução desta forma. Na laminação, é impossível obter o custo de uma solução gerada a partir de uma mudança na lista de prioridades de faixas de bitola, sem refazer a simulação da laminação de todas as ordens[1].

Apesar desse problema, e das várias ordenações de vetores necessárias na simulação citada, o programa conseguiu gerar, conforme visto no capítulo anterior, um grande número de soluções em pouco tempo de execução, graças a linguagem escolhida e ao fato de se utilizar extensivamente a memória virtual do micro.

Tais resultados permitem que seja recomendada, para uma possível continuidade do trabalho, a ampliação da complexidade do programa, com consideração de maiores

[1] Pelo menos não se conseguiu, apesar do esforço, elaborar um algoritmo para tanto.

detalhes do processo produtivo, tais como chegada de novas ordens durante a produção e existência de ordens em produção no instante inicial simulado.

Numa proposta mais simples, pode ser feita uma análise de sensibilidade do programa em relação aos custos utilizados, ou se estudar mais detalhadamente a influência da tabela de resfriamento nos resultados gerados.

BIBLIOGRAFIA

BUENO NETO, P. R. *Cadeias de Markov.* São Paulo. Apostila, Fundação Carlos Alberto Vanzolini.

BARRETO, M. R. P.; CHWIF, L. *A continual plane approach to the facility layout problem using Simulated Annealing.* São Paulo, 1994. Artigo, Departamento de Engenharia Mecânica, Escola Politécnica, Universidade de São Paulo.

EGLESE, R. W. 'Simulated Annealing: A tool for Operational Research'. *European Journal of Operational Researh,* v. 46, p. 271-281, 1990.

MORTON, T. E.; PENTICO, D. W. *Heuristic Scheduling Systems: with applications to production systems and project management.* Nova Iorque, John Wiley, 1993.

PRESS, W. H. *Numerical Recipes in C: The art of scientific computing.* 2.ed. Cambridge, Cambridge University Press, 1994.

SCHILDT, H. *C completo e total.* São Paulo, McGraw-Hill, 1991.

SCHILDT, H. *Linguagem C: guia do usuário.* São Paulo, McGraw-Hill, 1986.

VAN LAARHOVEN, P. J. M.; AARTS, E. H. L.; LENSTRA, J. K. 'Job shop scheduling by Simulated Annealing'. *Operations Research,* v. 46, p. 113-125, 1992.

ANEXO 1

```
/*      Escola Politecnica da Universidade de Sao Paulo
        Departamento de Engenharia de Producao
        Sequenciador.c: Software de programacao da producao para processo
        com set up e restricao de tamanho de lote utilizando o metodo
        de busca Simulated Annealing.
        Aluno: Paulo Henrique Barros Silva
        Orientador: Miguel Cezar Santoro
        Primeiro semestre de 1996                                        */


#include "STDLIB.H"
#include "STDIO.H"
#include "STRING.H"

#define TFACTR_FUND 0.92 /* Porcentagem para o calculo da proxima temperatura */
#define TFACTR_LAM  0.92  /* Idem, para o Simulated Annealing da laminacao */

#define TOT_ORDENS    682     /* Numero de ordens a serem processadas  */
#define TOT_GRUPOS    5       /* Quantidade de grupos de ligad diferentes */
#define TOT_FAIXAS    8       /* Quantidade de faixas de bitola diferentes */
#define MAX_FORNADAS  60      /* Numero maximo de fornadas             */
#define MAX_LOTELAM   70      /* Numero maximo de lotes de laminacao   */
#define ESPERA        36  /* Espera, apos o inico do funcionamento da fundicao, */
                          /* para o inicio do funcionamento da laminacao */

#define OK 'o'   /* defines utilizados em calcula_instante_final_lam */
#define ACABOU 'a'
#define QUEBRA 'q'

#define MBIG 1000000000      /* Defines para a funcao RAN3              */
#define MSEED 161803398      /* According to Knuth, any large MBIG, and */
#define MZ 0                 /* and any smaller (but still large) MSEED */
#define FAC (1.0/MBIG)       /* can be substited for the above values.  */



/* Definicao de estruturas */

struct grupo {
            int tp_fund;    /* tempo de fundicao do grupo em horas por fornada
*/
            float tp_desb;  /* tempo de desbaste do grupo em horas por 10 kg */
            int tp_acab[6]; /* tempo de acabamento em horas: 0 - bruto  */
            };                    /*                               1 - trefilado*/
                                  /*                               2 - descascado*/
                                  /*                               3 - retificado*/
                                  /*                               4 - torneado*/

struct faixa {
      int tp_setup_lam;   /* tempo de setup da faixa em horas */
      float tp_unit_lam;  /* tempo de processamento da faixa em horas por 10 kg
*/
        };

struct dados_ord {                  /* Dados obtidos do disco:           */
                int grupo;           /*  codigo do grupo  */
                char liga;            /* codigo da liga                    */
                int faixa;            /* codigo da faixa de bitola */
                int acab;             /* codigo do acabamento */
                int qtde;             /* quantidade em 10 kg      */
                int prazo_entr;       /* prazo de entrega em horas       */
                                   /* Dados gerados pelo programa:      */
             float fim_desb;  /*  instante de termino de desbaste em horas */
             float fim_lam;   /*  instante de termino de laminacao em horas */
             };


/* Estas estruturas sao utilizadas para a solucao da fundicao e para a
solucao da laminacao. */

struct fornada {
            struct dados_ord **inicio; /* inicio e fim apontam para ocorrencias
*/
            struct dados_ord **fim; /* de ordens_liga_folga[TOT_ORDENS],  */
```

```
            float folga_total; /* indicando inicio e fim de uma fornada. */
            };    /* folga total eh a soma das folgas de cada ordem de */
               /* fornada, multiplicada pela quantidade da mesma, utilizada na */
                           /* geracao de solucao semente para a fundicao. */

struct lote {
            struct dados_ord **inicio; /* inicio e fim apontam para ocorrencias
*/
            struct dados_ord **fim;  /* de ordens_faixa_fim_desb[TOT_ORDENS] */
            };                      /* indicando o inicio e o final de um lote */
                                     /* de laminacao. */



/* Prototipos das funcoes */

void Carrega_dados_do_disco(struct dados_ord ordens[TOT_ORDENS],
struct grupo tp_grupo[TOT_GRUPOS], struct faixa tp_faixa[TOT_FAIXAS],
int *po_lim_max, int *po_lim_min, int *po_ct_atraso,
int *po_ct_adiant, int *po_ct_estoque,
int *po_ct_setup_lam, int *po_ct_horizon);

void Gera_semente_fund(struct dados_ord ordens[TOT_ORDENS],
struct dados_ord *ordens_liga_folga[TOT_ORDENS],
struct grupo tp_grupo[TOT_GRUPOS], int lim_max, int lim_min,
struct faixa tp_faixa[TOT_FAIXAS], struct fornada sol_fund[MAX_FORNADAS], int
*po_num_fornadas);

void QS_folga(struct dados_ord *ordens_liga_folga[TOT_ORDENS], int left,
int right, struct grupo tp_grupo[TOT_GRUPOS], struct faixa tp_faixa[TOT_FAIXAS]);

void QS_folga_total(struct fornada sol_fund[MAX_FORNADAS], int left, int right);

void Agrupa_faixas(struct dados_ord ordens[TOT_ORDENS],
struct dados_ord *ordens_faixa_fim_desb[TOT_ORDENS],
struct dados_ord **inic_faixas[TOT_FAIXAS],
struct dados_ord **melh_faixas_seq_proc[TOT_FAIXAS]);

void Calcula_instante_final_desb(struct fornada sol_fund[MAX_FORNADAS],
int comeco_interv, int final_interv, struct grupo tp_grupo[TOT_GRUPOS]);

void Calcula_horizon_ideal(struct dados_ord ordens[TOT_ORDENS],
struct fornada sol_fund[MAX_FORNADAS], int num_fornadas,
struct grupo tp_grupo[TOT_GRUPOS],
struct faixa tp_faixa[TOT_FAIXAS],
 float *po_horizon_ideal_fund, float *po_horizon_ideal_lam);

void Gera_semente_lam(struct dados_ord *ordens_faixa_fim_desb[TOT_ORDENS],
struct dados_ord **inic_faixas[TOT_FAIXAS],
struct dados_ord **melh_faixas_seq_proc[TOT_FAIXAS],
struct dados_ord **faixas_seq_proc[TOT_FAIXAS],
struct grupo tp_grupo[TOT_GRUPOS],
struct faixa tp_faixa[TOT_FAIXAS],
struct lote sol_lam[MAX_LOTELAM],
struct dados_ord *ordens_seq_proc_lam[TOT_ORDENS]);

void QS_fim_desb(struct dados_ord *ordens_faixa_fim_desb[TOT_ORDENS],
int left, int right);

void Gera_vizinho_lam(struct dados_ord *ordens_faixa_fim_desb[TOT_ORDENS],
struct dados_ord **melh_parc_faixas_seq_proc[TOT_FAIXAS],
struct dados_ord **faixas_seq_proc[TOT_FAIXAS],
struct grupo tp_grupo[TOT_GRUPOS],
struct faixa tp_faixa[TOT_FAIXAS],
struct lote sol_lam[MAX_LOTELAM],
struct dados_ord *ordens_seq_proc_lam[TOT_ORDENS]);

void Calcula_instante_final_lam(
struct dados_ord *ordens_faixa_fim_desb[TOT_ORDENS],
struct dados_ord **faixas_seq_proc[TOT_FAIXAS],
struct grupo tp_grupo[TOT_GRUPOS], struct faixa tp_faixa[TOT_FAIXAS],
struct lote sol_lam[MAX_LOTELAM],
struct dados_ord *ordens_seq_proc_lam[TOT_ORDENS]);
```

```
char Pega_proximo(struct dados_ord **prox_ordem[TOT_FAIXAS], int *po_i,
float *po_tempo, struct faixa tp_faixa[TOT_FAIXAS]);

void QS_folga_lam(struct dados_ord *ordens_lote_folga[TOT_ORDENS], int left,
int right, struct grupo tp_grupo[TOT_GRUPOS], struct faixa tp_faixa[TOT_FAIXAS]);

float Calcula_custo(struct dados_ord ordens[TOT_ORDENS],
struct dados_ord *ordens_faixa_fim_desb[TOT_ORDENS],
struct lote sol_lam[MAX_LOTELAM],
struct dados_ord *ordens_seq_proc_lam[TOT_ORDENS],
struct grupo tp_grupo[TOT_GRUPOS], struct faixa tp_faixa[TOT_FAIXAS],
int ct_atraso, int ct_adiant, int ct_estoque,
int ct_setup_lam, int ct_horizon, float horizon_ideal_fund,
float horizon_ideal_lam);

int metrop(float de, float t);

void Gera_vizinho_fund(struct fornada melh_sol_fund[MAX_FORNADAS],
struct fornada sol_fund[MAX_FORNADAS], int num_fornadas, int *po_forn1,
int *po_forn2);

float ran3(long *idum);

double expp(double x);



void main()
{

FILE *arq_1custos, *arq_2custos;

register int i, j, l, m,  /* contadores para os Simulated Annealings */
             k,    /* contador */
nsuc_fund, nsuc_lam;  /* limites de iteracoes bem sucedidas em */
                      /* uma temperatura */
int pri_vez=1;

float t_fund, t_lam;  /* temperaturas do Simulated Annealing */

int forn1, forn2;  /* fornadas nos extremos da troca para obtencao de */
                   /* solucao vizinha na fundicao */

float horizon_ideal_fund,
      horizon_ideal_lam;

float custo,  /* custo da solucao para a qual se estuda ir (em R$) */
      custo_sem_lam;

/* Vetor de ordens de producao a serem processadas, agrupadas por liga */
struct dados_ord ordens[TOT_ORDENS];

/* Vetor de tempos de fundicao, desbaste e acabamento de cada liga  */
struct grupo tp_grupo[TOT_GRUPOS];

/* Vetor de tempos de setup e laminacao de cada faixa de bitola */
struct faixa tp_faixa[TOT_FAIXAS];

int lim_max,          /* capacidade do forno em 10 kg       */
    lim_min,          /* limite minimo do forno em 10 kg         */
    ct_atraso,        /* custos unitarios por hora de atraso, */
    ct_adiant,        /* adiantamento, estoque (R$/10kg.hora) e    */
    ct_estoque,       /* horizonte de processamento (R$/hora)    */
    ct_setup_lam,    /* e custo de set up da laminacao */
    ct_horizon;

/* Os dois vetores abaixo definem uma solucao para a fundicao. */
/* Vetor de ponteiros para ordens agrupado por liga e ordenado por
  folga dinamica. */
struct dados_ord *ordens_liga_folga[TOT_ORDENS];

/* Vetor de ponteiros para ordens_liga_folga indicando o inicio e o fim
  de cada fornada. A ordem do vetor define a ordem de processamento das
  fornadas.                                                              */
struct fornada sol_fund[MAX_FORNADAS];
```

```
int num_fornadas; /* Numero de fornadas necessarias para o processamento das
                     ordens. */

/* Vetor de ponteiros para ordens agrupado por faixa de bitola e
  ordenado por instante de termino de desbaste.                          */
struct dados_ord *ordens_faixa_fim_desb[TOT_ORDENS];

/* Vetor de ponteiros para ordens_faixa_fim_desb indicando o inicio das
  faixas.                                                                */
struct dados_ord **inic_faixas[TOT_FAIXAS];

/* Os quatro vetores abaixo definem uma solucao para a laminacao. */
/* Vetor de ponteiros para ordens_faixa_fim_desb indicando a sequencia de
  processamento das faixas.                                              */
struct dados_ord **faixas_seq_proc[TOT_FAIXAS];

/* Vetor de ponteiros para ordens_faixa_fim_desb indicando o inicio e o
  fim de cada lote de laminacao. A ordem do vetor define a ordem de
  processamento dos lotes.                                               */
struct lote sol_lam[MAX_LOTELAM];

/* Vetor de ponteiros para ordens agrupado por lote. Dentro de cada lote,
 a sequencia das ordens define a sequencia de laminacao. */
struct dados_ord *ordens_seq_proc_lam[TOT_ORDENS];

/* Vetores contendo a ultima solucao aceita. O vetor
ordens_liga_folga[TOT_ORDENS] faz parte da definacao da solucao, mas nao
precisa ser salvo porque eh sempre o mesmo. */

struct fornada melh_sol_fund[MAX_FORNADAS];

struct dados_ord **melh_faixas_seq_proc[TOT_FAIXAS];

struct lote melh_sol_lam[MAX_LOTELAM];

struct dados_ord *melh_ordens_seq_proc_lam[TOT_ORDENS];

float melh_custo = 3.3e+38;

/* Vetores contendo a ultima solucao para laminacao aceita
  para uma dada solucao para fundicao.                          */

struct dados_ord **melh_parc_faixas_seq_proc[TOT_FAIXAS];

struct lote melh_parc_sol_lam[MAX_LOTELAM];

struct dados_ord *melh_parc_ordens_seq_proc_lam[TOT_ORDENS];

float melh_parc_custo;

int aux, ct_nsuc_lam, ct_nsuc_fund;

arq_1custos=fopen("CUSTOS1.TXT", "w");
arq_2custos=fopen("CUSTOS2.TXT", "w");



Carrega_dados_do_disco(ordens, tp_grupo, tp_faixa,
&lim_max, &lim_min, &ct_atraso, &ct_adiant, &ct_estoque,
&ct_setup_lam, &ct_horizon);

Gera_semente_fund(ordens, ordens_liga_folga, tp_grupo, lim_max, lim_min,
tp_faixa, sol_fund, &num_fornadas);

Agrupa_faixas(ordens, ordens_faixa_fim_desb, inic_faixas, melh_faixas_seq_proc);

forn1 = 0;
forn2 = num_fornadas-1;
t_fund = 4.0e+4;
ct_nsuc_fund=0;
/* Simulated Annealing da fundicao. Passos de temperatura. */
for (i=1; i<=30; i++) {

    nsuc_fund=0;
```

```
/* Numero de buscas de solucao vizinha numa mesma temperatura. */
    for (j=1; j<=0.4*num_fornadas; j++) {

        Calcula_instante_final_desb(sol_fund, forn1, forn2, tp_grupo);

        if (pri_vez) {
           pri_vez=0;
           Calcula_horizon_ideal(ordens, sol_fund, num_fornadas,  tp_grupo,
                        tp_faixa, &horizon_ideal_fund, &horizon_ideal_lam);
        }

       Gera_semente_lam(ordens_faixa_fim_desb, inic_faixas,
                        melh_faixas_seq_proc, faixas_seq_proc, tp_grupo,
                        tp_faixa, sol_lam, ordens_seq_proc_lam);

        melh_parc_custo = Calcula_custo(ordens,
                    ordens_faixa_fim_desb, sol_lam, ordens_seq_proc_lam,
                    tp_grupo, tp_faixa, ct_atraso, ct_adiant,
                    ct_estoque, ct_setup_lam,
                    ct_horizon, horizon_ideal_fund,
                    horizon_ideal_lam);

/* Salva primeira solucao da laminacao como sendo a melhor solucao encontrada
ate o momento para uma dada solucao de fundicao */
       for (k=0; k<TOT_FAIXAS; k++)
            *(melh_parc_faixas_seq_proc+k) = *(faixas_seq_proc+k);

        for (k=0; (sol_lam+k)->inicio; k++)
            *(melh_parc_sol_lam+k) = *(sol_lam+k);

        (melh_parc_sol_lam+k)->inicio = NULL;

        for (k=0; k<TOT_ORDENS; k++)
            *(melh_parc_ordens_seq_proc_lam+k) = *(ordens_seq_proc_lam+k);

        fprintf(arq_lcustos,"\n%f\n",melh_parc_custo);
        custo_sem_lam=melh_parc_custo;

        t_lam = 3.0e+4;
        ct_nsuc_lam=0;

/* Simulated Annealing da laminacao. Passos de temperatura. */
        for (l=1; l<=25; l++) {

            nsuc_lam = 0;

/* Numero de buscas de solucao vizinha numa mesma temperatura. */
            for (m=1; m<=2*TOT_FAIXAS; m++) {

                Gera_vizinho_lam(ordens_faixa_fim_desb,
                                 melh_parc_faixas_seq_proc,
                                 faixas_seq_proc, tp_grupo, tp_faixa,
                                 sol_lam, ordens_seq_proc_lam);

                custo = Calcula_custo(ordens, ordens_faixa_fim_desb,
                                      sol_lam, ordens_seq_proc_lam, tp_grupo,
                                      tp_faixa, ct_atraso, ct_adiant, ct_estoque,
                                      ct_setup_lam, ct_horizon,
                                      horizon_ideal_fund, horizon_ideal_lam);

                if (metrop(custo - melh_parc_custo, t_lam)) {

/* Se a solucao eh melhor, ou o sorteio a favoreceu, deve ser salva. */
                    for (k=0; k<TOT_FAIXAS; k++)
                        *(melh_parc_faixas_seq_proc+k) = *(faixas_seq_proc+k);

                    for (k=0; (sol_lam+k)->inicio; k++)
                        *(melh_parc_sol_lam+k) = *(sol_lam+k);
```

```
                    (melh_parc_sol_lam+k)->inicio = NULL;

                    for (k=0; k<TOT_ORDENS; k++)
                        *(melh_parc_ordens_seq_proc_lam+k) =
                              *(ordens_seq_proc_lam+k);

                    melh_parc_custo = custo;

                    nsuc_lam++;

                    fprintf(arq_1custos,"%f\n",custo);

                }

/* Se houve muitos sucessos em uma temperatura, convem baixa-la. */
                if (nsuc_lam >= 0.4*TOT_FAIXAS) break;

            }

            t_lam *= TFACTR_LAM;

/* A busca eh encerrada caso nao seja aceita nenhuma solucao em um
passo de temperatura */
            if (nsuc_lam == 0) {
               ct_nsuc_lam++;
               if (ct_nsuc_lam>0) break;}
            else ct_nsuc_lam=0;

        }

/* Terminada a busca para a laminacao, verifica-se se esta solucao completa,
composta pela solucao vizinha da fundicao obtida e pela melhor solucao de
laminacao encontrada pelo Simulated Annealing da laminacao para ela, eh
melhor que a solucao completa anterior. */

        if (metrop(melh_parc_custo - melh_custo, t_fund)) {

/* Se a solucao eh melhor, ou o sorteio a favoreceu, deve ser salva. */
            for (k=0; (sol_fund+k)->inicio; k++)
                *(melh_sol_fund+k)=*(sol_fund+k);

            (melh_sol_fund+k)->inicio = NULL;

            for (k=0; k<TOT_FAIXAS; k++)
                *(melh_faixas_seq_proc+k) = *(melh_parc_faixas_seq_proc+k);

            for (k=0; (sol_lam+k)->inicio; k++)
                *(melh_sol_lam+k) = *(melh_parc_sol_lam+k);

            (melh_sol_lam+k)->inicio = NULL;

            for (k=0; k<TOT_ORDENS; k++)
                *(melh_ordens_seq_proc_lam+k) =
*(melh_parc_ordens_seq_proc_lam+k);

            melh_custo = melh_parc_custo;

            nsuc_fund++;

            fprintf(arq_2custos,"%f %f %f\n", custo_sem_lam, melh_parc_custo,
t_fund);

        }

        Gera_vizinho_fund(melh_sol_fund, sol_fund, num_fornadas,
                                                &forn1, &forn2);

/* Se houve muitos sucessos em uma temperatura, convem baixa-la. */
        if (nsuc_fund >= 0.15*num_fornadas) break;

    }

    t_fund *= TFACTR_FUND;
```

```
/* A busca eh encerrada caso nao seja aceita nenhuma solucao em
um passo de temperatura */
    if (nsuc_fund == 0) {
        ct_nsuc_fund++;
        if (ct_nsuc_fund>0) break;}
    else ct_nsuc_fund=0;

}



fclose(arq_1custos);
fclose(arq_2custos);




}



/* Leitura dos dados de entrada do programa em disco: ordens a serem
  processadas, tempos e restricoes de processamento e custos de atraso,
  adiantamento, estoque, set up e horizonte de processamento. Explicacao sobre os
  parametros da funcao: vide funcao main.                                    */

void Carrega_dados_do_disco(struct dados_ord ordens[TOT_ORDENS],
struct grupo tp_grupo[TOT_GRUPOS], struct faixa tp_faixa[TOT_FAIXAS],
int *po_lim_max, int *po_lim_min, int *po_ct_atraso,
int *po_ct_adiant, int *po_ct_estoque,
int *po_ct_setup_lam, int *po_ct_horizon)
{

FILE *arq_ordens, *arq_tp_grupo, *arq_tp_faixa, *arq_outros;
register int i;


/* Leitura das ordens a serem processadas. O programa pressupoe que estas
ordens ja estejam agrupadas por liga. */

arq_ordens=fopen("ORDENS", "r");
for (i=0; i<TOT_ORDENS; i++)
    fscanf(arq_ordens, "%d %c %d %d %d %d", &(ordens+i)->grupo,
      &(ordens+i)->liga, &(ordens+i)->faixa, &(ordens+i)->acab, &(ordens+i)-
>qtde,
      &(ordens+i)->prazo_entr);
fclose(arq_ordens);

/* Leitura dos tempos de fundicao, desbaste e acabamento de cada grupo-liga */

arq_tp_grupo=fopen("TP_GRUPO", "r");
for (i=0; i<TOT_GRUPOS; i++)
    fscanf(arq_tp_grupo, "%d %f %d %d %d %d %d %d", &((tp_grupo+i)->tp_fund),
                  &((tp_grupo+i)->tp_desb), &((tp_grupo+i)->tp_acab[0]),
                  &((tp_grupo+i)->tp_acab[1]),
                  &((tp_grupo+i)->tp_acab[2]),
                  &((tp_grupo+i)->tp_acab[3]),
                  &((tp_grupo+i)->tp_acab[4]),
                  &((tp_grupo+i)->tp_acab[5]));
fclose(arq_tp_grupo);

/* Leitura dos tempos de setup e laminacao de cada faixa de bitola */

arq_tp_faixa=fopen("TP_FAIXA", "r");
for (i=0; i<TOT_FAIXAS; i++)
    fscanf(arq_tp_faixa, "%d %f", &((tp_faixa+i)->tp_setup_lam),
                        &((tp_faixa+i)->tp_unit_lam));
fclose(arq_tp_faixa);

/* Leitura dos outros dados */

arq_outros=fopen("OUTROS", "r");
fscanf(arq_outros, "%d %d %d %d %d %d %d %d",
  po_lim_max, po_lim_min, po_ct_atraso, po_ct_adiant, po_ct_estoque,
  po_ct_setup_lam, po_ct_horizon);
```

```
fclose(arq_outros);

}



/* Gera solucao inicial ou semente da fundicao:
  Cria vetor de ponteiros para ordens agrupado por liga e ordenada por
  folga dinamica. Cria tambem vetor de ponteiros para o vetor anterior
  definindo as ordens que compoem cada fornada e a ordem de processamento
  das mesmas. Explicacao sobre os parametros da funcao: vide funcao main.  */

void Gera_semente_fund(struct dados_ord ordens[TOT_ORDENS],
struct dados_ord *ordens_liga_folga[TOT_ORDENS],
struct grupo tp_grupo[TOT_GRUPOS], int lim_max, int lim_min,
struct faixa tp_faixa[TOT_FAIXAS], struct fornada sol_fund[MAX_FORNADAS],
int *po_num_fornadas)
{

register int i,j,k;
int forn_qtde;
float aux_total;
struct dados_ord *aux_ordem;

/* Aponta para ordens. */
for (i=0; i<TOT_ORDENS; i++)
    *(ordens_liga_folga+i) = ordens+i;

/* Ordena os grupos de ordens de uma mesma liga por folga dinamica. */
for (i=0; i<TOT_ORDENS; i++) {
    for (j=i+1; j<TOT_ORDENS && (*(ordens_liga_folga+i))->liga ==
                        (*(ordens_liga_folga+j))->liga; j++);
    QS_folga(ordens_liga_folga, i, j-1, tp_grupo, tp_faixa);
    i=j-1;
}

/* Define fornadas. */

i=j=0;

/* Looping de cada liga. */
do {

/* Looping de cada fornada. */
   do {
      (sol_fund+j)->inicio = ordens_liga_folga+i;
      forn_qtde = 0;

      do {
       forn_qtde += (*(ordens_liga_folga+i))->qtde;
       i++;
      } while (i<TOT_ORDENS &&
           (*(ordens_liga_folga+i))->liga ==
           (*(ordens_liga_folga+i-1))->liga &&
           forn_qtde + (*(ordens_liga_folga+i))->qtde <= lim_max);

      (sol_fund+j)->fim = ordens_liga_folga+i-1;
      j++;
   } while (i<TOT_ORDENS &&
            (*(ordens_liga_folga+i))->liga ==
            (*(ordens_liga_folga+i-1))->liga);

   if (forn_qtde<lim_min) {
/* Se a ultima fornada de uma liga ficou com a quantidade abaixo do limite
minimo, deve se roubar ordens da fornada anterior. */

/* Rouba ordens da anterior ate que o limite minimo seja ultrapassado. */
      for (k=1; forn_qtde<lim_min; k++)
          forn_qtde += (*((sol_fund+j-1)->inicio-k))->qtde;

/* Acerta inicio e fim dos dois ultimos lotes da liga. */
      (sol_fund+j-2)->fim = ((sol_fund+j-1)->inicio)-(k+1);
      (sol_fund+j-1)->inicio = ((sol_fund+j-1)->inicio)-k;

/* Ve se a penultima fornada nao ficou com quantidade abaixo do minimo. */
```

```
      for (k=0, forn_qtde=0; k < (sol_fund+j-2)->fim+1 - (sol_fund+j-2)->inicio;
                  k++) forn_qtde += (*((sol_fund+j-2)->inicio)+k)->qtde;
      if (forn_qtde < lim_min) {
         printf("Problema na definicao das fornadas");
         exit(0); }
   }
} while (i<TOT_ORDENS);

*po_num_fornadas = j;
(sol_fund+j)->inicio = NULL;


/* Calcula folga total de cada fornada. */

for (i=0; (sol_fund+i)->inicio; i++) {
    for (j=0, aux_total=0; (sol_fund+i)->inicio+j<=(sol_fund+i)->fim; j++) {
        aux_ordem = *((sol_fund+i)->inicio+j);
        aux_total += (aux_ordem->prazo_entr - (tp_grupo+aux_ordem->grupo)-
>tp_fund -
        aux_ordem->qtde *  ( (tp_faixa+aux_ordem->faixa)->tp_unit_lam +
        (tp_grupo+aux_ordem->grupo)->tp_desb ) -(tp_grupo
        +aux_ordem->grupo)->tp_acab[aux_ordem->acab]) * aux_ordem->qtde;
    }
    (sol_fund+i)->folga_total=aux_total;
}

/* Ordena sol_fund por folga_total. */
QS_folga_total(sol_fund, 0, *po_num_fornadas-1);

}



/* Quick sort recursivo por folga dinamica na fundicao.
ordens_liga_folga[TOT_ORDENS]: vetor que sera ordenado.
left, right: delimitadores do trecho do vetor que deve ser ordenado.
tp_grupo, tp_faixa: vetores com dados para comparacao entre os elementos. */

void QS_folga(struct dados_ord *ordens_liga_folga[TOT_ORDENS], int left,
int right, struct grupo tp_grupo[TOT_GRUPOS], struct faixa tp_faixa[TOT_FAIXAS])
{

register int i,j;
float folga_do_meio, folga_aux;
struct dados_ord *y, *meio;

i=left;
j=right;

meio = *(ordens_liga_folga+(left+right)/2);
folga_do_meio = (meio->prazo_entr - (tp_grupo+meio->grupo)->tp_fund -meio->qtde *
   ((tp_faixa+meio->faixa)->tp_unit_lam+(tp_grupo+meio->grupo)->tp_desb)
   - (tp_grupo+meio->grupo)->tp_acab[meio->acab]) * meio->qtde;

do {

   folga_aux = ((*(ordens_liga_folga+i))->prazo_entr -
      (tp_grupo+((*(ordens_liga_folga+i))->grupo))->tp_fund -
      (*(ordens_liga_folga+i))->qtde *
      ( (tp_faixa+(*(ordens_liga_folga+i))->faixa)->tp_unit_lam +
      (tp_grupo+(*(ordens_liga_folga+i))->grupo)->tp_desb)-(tp_grupo
      +(*(ordens_liga_folga+i))->grupo)->tp_acab[(*(ordens_liga_folga+i))->acab])
      * (*(ordens_liga_folga+i))->qtde;
   while (folga_aux < folga_do_meio &&
               i < right) {
         i++;
         folga_aux = ((*(ordens_liga_folga+i))->prazo_entr -
           (tp_grupo+((*(ordens_liga_folga+i))->grupo))->tp_fund -
           (*(ordens_liga_folga+i))->qtde *
           ( (tp_faixa+(*(ordens_liga_folga+i))->faixa)->tp_unit_lam +
           (tp_grupo+(*(ordens_liga_folga+i))->grupo)->tp_desb)-(tp_grupo
           +(*(ordens_liga_folga+i))->grupo)->tp_acab[(*(ordens_liga_folga+i))-
>acab])
           * (*(ordens_liga_folga+i))->qtde;
   }
```

```
   folga_aux = ((*(ordens_liga_folga+j))->prazo_entr -
      (tp_grupo+((*(ordens_liga_folga+j))->grupo))->tp_fund -
      (*(ordens_liga_folga+j))->qtde *
      ( (tp_faixa+(*(ordens_liga_folga+j))->faixa)->tp_unit_lam +
      (tp_grupo+(*(ordens_liga_folga+j))->grupo)->tp_desb)-(tp_grupo
      +(*(ordens_liga_folga+j))->grupo)->tp_acab[(*(ordens_liga_folga+j))->acab])
      * (*(ordens_liga_folga+j))->qtde;
   while (folga_do_meio < folga_aux && j > left) {
         j--;
         folga_aux = ((*(ordens_liga_folga+j))->prazo_entr -
         (tp_grupo+((*(ordens_liga_folga+j))->grupo))->tp_fund -
         (*(ordens_liga_folga+j))->qtde *
         ( (tp_faixa+(*(ordens_liga_folga+j))->faixa)->tp_unit_lam +
         (tp_grupo+(*(ordens_liga_folga+j))->grupo)->tp_desb)-(tp_grupo
         +(*(ordens_liga_folga+j))->grupo)->tp_acab[(*(ordens_liga_folga+j))-
>acab])
         * (*(ordens_liga_folga+j))->qtde;
   }

   if (i<=j) {
      y = *(ordens_liga_folga+i);
      *(ordens_liga_folga+i) = *(ordens_liga_folga+j);
      *(ordens_liga_folga+j) = y;
      i++;
      j--;
   }
} while (i<=j);
if (left<j) QS_folga(ordens_liga_folga, left, j, tp_grupo, tp_faixa);
if (i<right) QS_folga(ordens_liga_folga, i, right, tp_grupo, tp_faixa);

}



/* Quick sort recursivo por folga total da fornada.
sol_fund[MAX_FORNADAS]: vetor que sera ordenado.
left, right: delimitadores do trecho do vetor que deve ser ordenado.  */

void QS_folga_total(struct fornada sol_fund[MAX_FORNADAS], int left, int right)
{

register int i,j;
float folga_total_do_meio;
struct fornada y;

i=left;
j=right;

folga_total_do_meio = (sol_fund + (left+right)/2)->folga_total;

do {

   while ((sol_fund+i)->folga_total < folga_total_do_meio && i < right) i++;
   while (folga_total_do_meio < (sol_fund+j)->folga_total && j > left) j--;

   if (i<=j) {
      y = *(sol_fund+i);
      *(sol_fund+i) = *(sol_fund+j);
      *(sol_fund+j) = y;
      i++;
      j--;
   }
} while (i<=j);
if (left<j) QS_folga_total(sol_fund, left, j);
if (i<right) QS_folga_total(sol_fund, i, right);

}



/* Cria vetor de ponteiros para ordens agrupado por faixas de bitola.
  Cria tambem vetor de ponteiros para o vetor anterior indicando o comeco de
cada faixa.
ordens_faixa_fim_desb: vetor cujas ocorrencias serao agrupadas.
```

```
Explicacao sobre os demais parametros: vide funcao main.   */

void Agrupa_faixas(struct dados_ord ordens[TOT_ORDENS],
struct dados_ord *ordens_faixa_fim_desb[TOT_ORDENS],
struct dados_ord **inic_faixas[TOT_FAIXAS],
struct dados_ord **melh_faixas_seq_proc[TOT_FAIXAS])
{

register int i,j,k,l;

struct dados_ord *x;

/* Copia enderecos de ordens. */
for (i=0; i<TOT_ORDENS; i++)
    *(ordens_faixa_fim_desb+i) = ordens+i;

/* Looping do agrupamento. */
for (i=0, l=0; i<TOT_ORDENS; i++) {

/* Guarda o inicio de uma faixa (tambem no vetor que define a sequencia de
processamento das faixas). */
    *(inic_faixas+l) = ordens_faixa_fim_desb+i;
    *(melh_faixas_seq_proc+l) = ordens_faixa_fim_desb+i;
    l++;

/* Procura a proxima ocorrencia com faixa diferente. */
    for (j=i+1; j<TOT_ORDENS && (*(ordens_faixa_fim_desb+i))->faixa ==
        (*(ordens_faixa_fim_desb+j))->faixa; j++);
    if (j>=TOT_ORDENS) break;

    k=j+1;

/* Ate o fim do vetor. */
    for (;;) {

/* Procura agora a proxima ocorrencia da mesma faixa. */
        while (k<TOT_ORDENS && (*(ordens_faixa_fim_desb+k))->faixa !=
                                    (*(ordens_faixa_fim_desb+i))->faixa) k++;
        if (k >= TOT_ORDENS) {
           i=j-1;
           break;
       }
/* Troca as duas (faixa diferente por mesma faixa). */
       x = *(ordens_faixa_fim_desb+j);
       *(ordens_faixa_fim_desb+j) = *(ordens_faixa_fim_desb+k);
       *(ordens_faixa_fim_desb+k) = x;
        j=j+1;
    }
}

}



/* Calcula o instante de termino de desbaste para uma dada solucao.
  Comeco_interv e final_interv sao os extremos do intervalo de sol_fund onde
  o processamento precisa ser feito.                                    */

void Calcula_instante_final_desb(struct fornada sol_fund[MAX_FORNADAS],
 int comeco_interv, int final_interv, struct grupo tp_grupo[TOT_GRUPOS])
{

register int i,j;
/* tempo -> instante de termino da fornada. */
int tempo;

/* Inicialmente, calcula o instante de termino da ultima fornada. */
if (comeco_interv == 0) tempo=0;
else tempo = (*((sol_fund+comeco_interv-1)->inicio ))->fim_desb -
          ( (*((sol_fund+comeco_interv-1)->inicio ))->qtde *
         (tp_grupo+(*((sol_fund+comeco_interv-1)->inicio))->grupo)->tp_desb);

for (i=comeco_interv; i<=final_interv; i++) {

/* Calcula o instante de termino da fornada atual. */
```

```
    tempo += (tp_grupo+(*((sol_fund+i)->inicio))->grupo)->tp_fund;

    for (j=0; j < (sol_fund+i)->fim - (sol_fund+i)->inicio + 1; j++)
       (*((sol_fund+i)->inicio + j))->fim_desb = tempo +
                    (*((sol_fund+i)->inicio + j))->qtde *
             (tp_grupo+(*((sol_fund+i)->inicio + j))->grupo)->tp_desb;
}

}



/* Calcula horizontes ideais para fundicao e laminacao. */

void Calcula_horizon_ideal(struct dados_ord ordens[TOT_ORDENS],
struct fornada sol_fund[MAX_FORNADAS], int num_fornadas,
struct grupo tp_grupo[TOT_GRUPOS],
struct faixa tp_faixa[TOT_FAIXAS],
float *po_horizon_ideal_fund, float *po_horizon_ideal_lam)
{


register int i;
float tp_lam = ESPERA;
int aux_acab;
struct dados_ord *aux_fim;

/* Fundicao. */

aux_fim = *((sol_fund+num_fornadas-1)->fim);
aux_acab = aux_fim->acab;
*po_horizon_ideal_fund = aux_fim->fim_desb +
                 (tp_faixa+aux_fim->faixa)->tp_unit_lam * aux_fim->qtde +
                 (tp_grupo+aux_fim->grupo)->tp_acab[aux_acab];

/* Laminacao. */

for (i=0, tp_lam=0; i<TOT_ORDENS; i++)
    tp_lam += ((ordens+i)->qtde) * (tp_faixa+(ordens+i)->faixa)->tp_unit_lam;

for (i=0; i<TOT_FAIXAS; i++)
    tp_lam += (tp_faixa+i)->tp_setup_lam;

tp_lam += (tp_grupo+aux_fim->grupo)->tp_acab[aux_acab];

*po_horizon_ideal_lam = tp_lam;

}



/* Gera solucao inicial ou semente da laminacao para uma dada solucao
  na fundicao:
    Cria vetor de sequencia de processamento das faixas e ordena vetor de
  ponteiros para ordens (ordens_faixa_fim_desb), ja agrupado por faixa, por
  instante de termino de desbaste. A partir destes duas vetores, cria
  vetor de ponteiros para o ultimo definindo os lotes de ordens e a ordem
  em que serao laminados. Em seguida, cria vetor de ponteiros para ordens
  definindo a sequencia de processamento das ordens em cada lote. Explicacao
  sobre os parametros  da funcao: vide funcao main.                    */

void Gera_semente_lam(struct dados_ord *ordens_faixa_fim_desb[TOT_ORDENS],
struct dados_ord **inic_faixas[TOT_FAIXAS],
struct dados_ord **melh_faixas_seq_proc[TOT_FAIXAS],
struct dados_ord **faixas_seq_proc[TOT_FAIXAS],
struct grupo tp_grupo[TOT_GRUPOS],
struct faixa tp_faixa[TOT_FAIXAS],
struct lote sol_lam[MAX_LOTELAM],
struct dados_ord *ordens_seq_proc_lam[TOT_ORDENS])
{

register int i;

/* Ordena vetor de ponteiros para ordens, ja agrupado por faixa, por
  instante de termino de desbaste.                                    */
```

```
for (i=0; i<TOT_FAIXAS-1; i++)
    QS_fim_desb(ordens_faixa_fim_desb, *(inic_faixas+i) - ordens_faixa_fim_desb,
                                       *(inic_faixas+i+1) - ordens_faixa_fim_desb -
1);
QS_fim_desb(ordens_faixa_fim_desb, *(inic_faixas+TOT_FAIXAS-1) -
                            ordens_faixa_fim_desb, TOT_ORDENS-1);

/* Cria vetor de sequencia de processamento das faixas, baseado na melhor
sequencia de processamento da solucao para fundicao anterior. */

for (i=0; i<TOT_FAIXAS; i++)
    *(faixas_seq_proc+i) = *(melh_faixas_seq_proc+i);

Calcula_instante_final_lam(ordens_faixa_fim_desb, faixas_seq_proc,
                           tp_grupo, tp_faixa, sol_lam, ordens_seq_proc_lam);

}



/* Quick sort por instante de termino de desbaste
ordens_faixa_fim_desb[TOT_ORDENS]: vetor que sera ordenado.
left, right: delimitadores do trecho do vetor que deve ser ordenado. */

void QS_fim_desb(struct dados_ord *ordens_faixa_fim_desb[TOT_ORDENS],
int left, int right)

{
register int i,j;
float fim_desb_do_meio, fim_desb_aux;
struct dados_ord *y;

i=left;
j=right;
fim_desb_do_meio = (*(ordens_faixa_fim_desb+(left+right)/2))->fim_desb;

do {
   while ((*(ordens_faixa_fim_desb+i))->fim_desb < fim_desb_do_meio &&
          i < right) i++;
   while (fim_desb_do_meio < (*(ordens_faixa_fim_desb+j))->fim_desb &&
          j > left) j--;
   if (i<=j) {
      y = *(ordens_faixa_fim_desb+i);
      *(ordens_faixa_fim_desb+i) = *(ordens_faixa_fim_desb+j);
      *(ordens_faixa_fim_desb+j) = y;
      i++;
      j--;
   }
} while (i<=j);
if (left<j) QS_fim_desb(ordens_faixa_fim_desb, left, j);
if (i<right) QS_fim_desb(ordens_faixa_fim_desb, i, right);

}



/* Gera solucao vizinha a solucao atual parcial na laminacao:
   Troca duas faixas de posicao na sequencia de processamento das faixas.
  Baseada nessa sequencia, cria vetor de ponteiros para ordens definindo
  os lotes de ordens e a ordem em que serao laminados. Em seguida, cria
  vetor de ponteiros para ordens definindo a sequencia de processamento
  das ordens em cada lote. Explicacao sobre os parametros da funcao: vide
  funcao main. */

void Gera_vizinho_lam(struct dados_ord *ordens_faixa_fim_desb[TOT_ORDENS],
struct dados_ord **melh_parc_faixas_seq_proc[TOT_FAIXAS],
struct dados_ord **faixas_seq_proc[TOT_FAIXAS],
struct grupo tp_grupo[TOT_GRUPOS],
struct faixa tp_faixa[TOT_FAIXAS],
struct lote sol_lam[MAX_LOTELAM],
struct dados_ord *ordens_seq_proc_lam[TOT_ORDENS])
{

register int i;
```

```
static long idum=-1;
int faix1, faix2, aux;
float aux_ran;
struct dados_ord **aux_troca;

/* A geracao de vizinho eh baseada na melhor solucao da laminacao ate o
momento para a atual solucao da fundicao. */
for (i=0; i<TOT_FAIXAS; i++)
    *(faixas_seq_proc+i) = *(melh_parc_faixas_seq_proc+i);

/* Sorteia duas faixas para troca de posicao na sequencia de processamento. */
do {
   aux_ran = ran3(&idum);
} while ((int) aux_ran == 1);
faix1 = (int) (TOT_FAIXAS*aux_ran);

do {
   do {
      aux_ran = ran3(&idum);
   } while ((int) aux_ran==1);
   faix2 = (int) ((TOT_FAIXAS-1)*aux_ran);
} while ((faix2-faix1)>1 || (faix1-faix2)>2); /* nao permite sortear */
                                    /* trechos muito grandes (maximo 3). */

if (faix2 >= faix1) faix2++;
else {
   aux=faix1;
   faix1=faix2;
   faix2=aux;
}


/* Troca faixas. */

for (i=faix1; i<(faix1+faix2+1)/2;i++) {
    aux_troca = *(faixas_seq_proc+i);
    *(faixas_seq_proc+i) = *(faixas_seq_proc+faix2-(i-faix1));
    *(faixas_seq_proc+faix2-(i-faix1)) = aux_troca;
}

Calcula_instante_final_lam(ordens_faixa_fim_desb, faixas_seq_proc,
tp_grupo, tp_faixa, sol_lam, ordens_seq_proc_lam);

}



void Calcula_instante_final_lam(
struct dados_ord *ordens_faixa_fim_desb[TOT_ORDENS],
struct dados_ord **faixas_seq_proc[TOT_FAIXAS],
struct grupo tp_grupo[TOT_GRUPOS],  struct faixa tp_faixa[TOT_FAIXAS],
struct lote sol_lam[MAX_LOTELAM],
struct dados_ord *ordens_seq_proc_lam[TOT_ORDENS])
{

register int i,
    j,k;
float tempo;
int par_i,
    ainda_nao_foi;
char ret;

/* Indicador de processamento ja realizado em ordens_lote_folga. */
struct dados_ord lixo;

/* Vetor de ponteiros para ordens_faixa_fim_desb que indica a ordem de
  producao de cada faixa que esta sendo processada. */
struct dados_ord **prox_ordem[TOT_FAIXAS];

/* Vetor de ponteiros para ordens agrupado por lote de laminacao e ordenado
  por folga dinamica na laminacao. A ordem dos lotes de laminacao no vetor
  eh determinada pela ordem das faixas no vetor ordens_faixa_fim_desb.  */
struct dados_ord *ordens_lote_folga[TOT_ORDENS];

/* Aponta inicio das faixas. */
```

```
for (i=0; i<TOT_FAIXAS; i++)
    *(prox_ordem+i) = *(faixas_seq_proc+i);

tempo=ESPERA;

/* Seleciona disponivel no instante atual. */

for (i=0; i<TOT_FAIXAS; i++)
    if ((**(prox_ordem+i))->fim_desb <=
         tempo+(tp_faixa+(**(prox_ordem+i))->faixa)->tp_setup_lam) break;
if (i>=TOT_FAIXAS) {

/* Nao tem nenhum disponivel (as filas estao vazias). Deve-se procurar o que
vai chegar primeiro. */
   int menor_fim_desb=0;
   for (i=1; i<TOT_FAIXAS; i++)
       if ((**(prox_ordem+i))->fim_desb <
            (**(prox_ordem+menor_fim_desb))->fim_desb) menor_fim_desb=i;
   tempo = (**(prox_ordem+menor_fim_desb))->fim_desb -
           (tp_faixa+(**(prox_ordem+menor_fim_desb))->faixa)->tp_setup_lam;
   i=menor_fim_desb;
}


/* Define lotes para laminacao. */

j=-1;

do {
/* Inicio de um lote. */
   j++;
   (sol_lam+j)->inicio=*(prox_ordem+i);
   tempo+= (tp_faixa+(**(prox_ordem+i))->faixa)->tp_setup_lam;
   do {

/* Coloca o selecionado no final do lote. */
      (sol_lam+j)->fim=*(prox_ordem+i);
      tempo += (**(prox_ordem+i))->qtde *
               (tp_faixa+(**(prox_ordem+i))->faixa)->tp_unit_lam;
/* Avanca previamente prox_ordem. */
      if ((*(prox_ordem+i))+1 - ordens_faixa_fim_desb >= TOT_ORDENS ||
          (**(prox_ordem+i))->faixa !=  (*((*(prox_ordem+i))+1))->faixa)
         *(prox_ordem+i)=NULL;
      else *(prox_ordem+i) = (*(prox_ordem+i))+1;
      par_i=i;
/* Seleciona outro para entrar no lote. */
      ret = Pega_proximo(prox_ordem, &par_i, &tempo, tp_faixa);
      i=par_i;
   } while (ret!=ACABOU && ret!=QUEBRA);
} while (ret!=ACABOU);

if (j+1>MAX_LOTELAM) {printf("Aumentar MAX_LOTELAM."); exit(0);}

(sol_lam+j+1)->inicio=NULL;

/* Cria vetor de ponteiros para ordens agrupado por faixa (identicamente a
ordens_folga_fim_desb) e ordenado por folga dinamica na laminacao.   */

for (i=0; i<TOT_ORDENS; i++)
    *(ordens_lote_folga+i) = *(ordens_faixa_fim_desb+i);

for (i=0; (sol_lam+i)->inicio; i++)
    QS_folga_lam(ordens_lote_folga, (sol_lam+i)->inicio -
                 ordens_faixa_fim_desb, (sol_lam+i)->fim -
              ordens_faixa_fim_desb, tp_grupo, tp_faixa);

/* Cria vetor de ponteiros para ordens definindo a sequencia de
  processamento das ordens em cada lote.                                  */

tempo=ESPERA;

/* Looping ate o fim dos lotes. */
for (i=0; (sol_lam+i)->inicio; i++) {
    tempo += (tp_faixa+(*((sol_lam+i)->inicio))->faixa)->tp_setup_lam;
```

```
/* Looping ate o fim deste lote. */
    for (ainda_nao_foi = (sol_lam+i)->inicio - ordens_faixa_fim_desb, k=0 ;
        ainda_nao_foi <= (sol_lam+i)->fim - ordens_faixa_fim_desb;) {

/* Pega primeiro na fila disponivel no instante atual (estao por ordem de folga
dinamica). */
        for (j=ainda_nao_foi; j <= (sol_lam+i)->fim - ordens_faixa_fim_desb;
             j++) {
            if (*(ordens_lote_folga+j) == &lixo) continue;
            if ((*(ordens_lote_folga+j))->fim_desb <= tempo) {
             *(ordens_seq_proc_lam + ((sol_lam+i)->inicio
               - ordens_faixa_fim_desb + k)) = *(ordens_lote_folga+j);

/* Sera o proximo a ser laminado. */
               k++;
               tempo += (*(ordens_lote_folga+j))->qtde *
                      (tp_faixa+(*(ordens_lote_folga+j))->faixa)->tp_unit_lam;
               (*(ordens_lote_folga+j))->fim_lam = tempo;
               *(ordens_lote_folga+j) = &lixo;
               break;
            }
        }

        if (j > (sol_lam+i)->fim - ordens_faixa_fim_desb) {

/* Por nao ter nenhum disponivel no instante atual, chegou ao final do lote.
Procura agora a ordem deste lote que vai chega primeiro do desbaste. */
           int menor_fim_desb=ainda_nao_foi;
           for (j=ainda_nao_foi; j <= (sol_lam+i)->fim - ordens_faixa_fim_desb;
                j++) {
               if (*(ordens_lote_folga+j) == &lixo) continue;
               if ((*(ordens_lote_folga+j))->fim_desb <
                    (*(ordens_lote_folga + menor_fim_desb))->fim_desb)
                  menor_fim_desb=j;
           }

/* E a inclui na solucao como proxima a ser laminada. */
           *(ordens_seq_proc_lam + ((sol_lam+i)->inicio - ordens_faixa_fim_desb
                + k)) = *(ordens_lote_folga + menor_fim_desb);
           k++;
           tempo = (*(ordens_lote_folga + menor_fim_desb))->fim_desb +
             (*(ordens_lote_folga + menor_fim_desb))->qtde *
             (tp_faixa+(*(ordens_lote_folga + menor_fim_desb))->faixa)-
>tp_unit_lam;
           (*(ordens_lote_folga + menor_fim_desb))->fim_lam = tempo;
           *(ordens_lote_folga+menor_fim_desb) = &lixo;
           j = menor_fim_desb;
        }

/* Deve continuar o looping. Se a ordem incluida eh a primeira, avanca pulando
as ja incluidas. */
        if (j == ainda_nao_foi)
           for (; ainda_nao_foi <= (sol_lam+i)->fim-ordens_faixa_fim_desb;
                ainda_nao_foi++)
             if (*(ordens_lote_folga+ainda_nao_foi) != &lixo)
                break;
    }
}

}

/* Posiciona na proxima ordem a ser adicionada na solucao. */

char Pega_proximo(struct dados_ord **prox_ordem[TOT_FAIXAS], int *po_i,
float *po_tempo, struct faixa tp_faixa[TOT_FAIXAS])
{

register int i=*po_i;
int deu_a_volta=0,
       menor_fim_desb;

if (*(prox_ordem+i))
/* Faixa com ordens ainda nao incluidas na solucao. Se fim_desb<=tempo,
 eh o proximo (o selecionado). */
```

```
   if ((**(prox_ordem+i))->fim_desb <= *po_tempo) return OK;
   else {
/*  Ja que nao eh o proximo, procura uma faixa nao terminada. */
      for (i=i+1; i<TOT_FAIXAS; i++)
          if ((*(prox_ordem+i))) break;
      if (i>=TOT_FAIXAS) {
          for (i=0; i<*po_i; i++)
              if ((*(prox_ordem+i))) break;
          if (i>=*po_i) {
/* Nao achou. Esta eh a unica nao terminada. */
              *po_tempo=(**(prox_ordem+*po_i))->fim_desb;
              return OK; /* Continua o mesmo lote. */
          }
      }
   }
else {
/* Eh fim de faixa (todas as ordens ja incluidas). Procura uma faixa nao
 terminada. */
   for (i=i+1; i<TOT_FAIXAS; i++)
       if ((*(prox_ordem+i))) break;
   if (i>=TOT_FAIXAS) {
      for (i=0; i<*po_i; i++)
          if ((*(prox_ordem+i))) break;
      if (i>=*po_i)
/* Nao tem nenhuma. */
         return ACABOU;
   }
}


/* Procura uma faixa que tenha ordem, nao incluida num lote, na fila
 no instante atual. */
while (!deu_a_volta) {
/* Esta disponivel ? */
   if ((**(prox_ordem+i))->fim_desb <=
       *po_tempo + (tp_faixa+(**(prox_ordem+i))->faixa)->tp_setup_lam) {
/* Sim. Comeca novo lote. */
       *po_i=i;
       return QUEBRA;
   }
/* Nao esta disponivel. Procura faixa nao terminada. */
   if (i>=*po_i) {
      for (i=i+1; i<TOT_FAIXAS; i++)
          if ((*(prox_ordem+i))) break;
      if (i>=TOT_FAIXAS) {
         for (i=0; i<*po_i; i++)
             if ((*(prox_ordem+i))) break;
         if (i>=*po_i) deu_a_volta=1; /* Nao achou.*/
      }
   }
   else {
      for (i=i+1; i<*po_i; i++)
          if ((*(prox_ordem+i))) break;
      if (i>=*po_i) deu_a_volta=1; /* Nao achou.*/
   }
}

/* Nao achou nenhuma faixa com ordem, nao incluida num lote, na fila no
instante atual. Procura a que vai chegar primeiro na fila. */
for (i=0; !(*(prox_ordem+i)); i++);
menor_fim_desb=i;
for (i=i+1; i<TOT_FAIXAS; i++)
       if ((*(prox_ordem+i)) && (**(prox_ordem+i))->fim_desb <
              (**(prox_ordem+menor_fim_desb))->fim_desb)
             menor_fim_desb=i;
if (menor_fim_desb == *po_i) {
   *po_tempo = (**(prox_ordem+menor_fim_desb))->fim_desb;
   return OK; /* Eh da mesma faixa. Nao comeca outro lote. */
}
else {
   *po_tempo = (**(prox_ordem+menor_fim_desb))->fim_desb -
               (tp_faixa+(**(prox_ordem+menor_fim_desb))->faixa)->tp_setup_lam;
   *po_i = menor_fim_desb;
   return QUEBRA; /* Eh de outra faixa. Comeca outro lote. */
}
```

```
}


/* Quick sort recursivo por folga dinamica na laminacao.
ordens_lote_folga[TOT_ORDENS]: vetor que sera ordenado.
left, right: delimitadores do trecho do vetor que deve ser ordenado.
tp_grupo, tp_faixa: vetores com dados para comparacao entre as ocorrencias. */

void QS_folga_lam(struct dados_ord *ordens_lote_folga[TOT_ORDENS], int left,
int right, struct grupo tp_grupo[TOT_GRUPOS], struct faixa tp_faixa[TOT_FAIXAS])
{

register int i,j;
float folga_do_meio, folga_aux;
struct dados_ord *y, *meio;

i=left;
j=right;
meio=*(ordens_lote_folga+(left+right)/2);
folga_do_meio = (meio->prazo_entr -
                meio->qtde * (tp_faixa+meio->faixa)->tp_unit_lam -
                (tp_grupo+meio->grupo)->tp_acab[meio->acab])
                * meio->qtde;

do {

   folga_aux=((*(ordens_lote_folga+i))->prazo_entr -
          (*(ordens_lote_folga+i))->qtde *
          (tp_faixa+(*(ordens_lote_folga+i))->faixa)->tp_unit_lam - (tp_grupo+
          (*(ordens_lote_folga+i))->grupo)->tp_acab[(*(ordens_lote_folga+i))-
>acab])
             * (*(ordens_lote_folga+i))->qtde;
   while (folga_aux < folga_do_meio &&
                  i < right) {
       i++;
       folga_aux=((*(ordens_lote_folga+i))->prazo_entr -
          (*(ordens_lote_folga+i))->qtde *
          (tp_faixa+(*(ordens_lote_folga+i))->faixa)->tp_unit_lam - (tp_grupo+
          (*(ordens_lote_folga+i))->grupo)->tp_acab[(*(ordens_lote_folga+i))-
>acab])
                  * (*(ordens_lote_folga+i))->qtde;
   }

   folga_aux=((*(ordens_lote_folga+j))->prazo_entr -
         (*(ordens_lote_folga+j))->qtde *
         (tp_faixa+(*(ordens_lote_folga+j))->faixa)->tp_unit_lam - (tp_grupo+
         (*(ordens_lote_folga+j))->grupo)->tp_acab[(*(ordens_lote_folga+j))-
>acab])
              * (*(ordens_lote_folga+j))->qtde;
   while (folga_do_meio < folga_aux
                  && j > left) {
       j--;
       folga_aux=((*(ordens_lote_folga+j))->prazo_entr -
          (*(ordens_lote_folga+j))->qtde *
          (tp_faixa+(*(ordens_lote_folga+j))->faixa)->tp_unit_lam - (tp_grupo+
          (*(ordens_lote_folga+j))->grupo)->tp_acab[(*(ordens_lote_folga+j))-
>acab])
                * (*(ordens_lote_folga+j))->qtde;
   }

   if (i<=j) {
      y = *(ordens_lote_folga+i);
      *(ordens_lote_folga+i) = *(ordens_lote_folga+j);
      *(ordens_lote_folga+j) = y;
      i++;
      j--;
   }
} while (i<=j);
if (left<j) QS_folga_lam(ordens_lote_folga, left, j, tp_grupo, tp_faixa);
if (i<right) QS_folga_lam(ordens_lote_folga, i, right, tp_grupo, tp_faixa);

}
```

```
/* Calcula custos de atraso ou adiantamento, estoque, set up e horizonte de
  planejamento de uma dada solucao. */

float Calcula_custo(struct dados_ord ordens[TOT_ORDENS],
struct dados_ord *ordens_faixa_fim_desb[TOT_ORDENS],
struct lote sol_lam[MAX_LOTELAM],
struct dados_ord *ordens_seq_proc_lam[TOT_ORDENS],
struct grupo tp_grupo[TOT_GRUPOS],
struct faixa tp_faixa[TOT_FAIXAS],
int ct_atraso, int ct_adiant, int ct_estoque,
int ct_setup_lam, int ct_horizon, float horizon_ideal_fund,
float horizon_ideal_lam)
{

register int i;
float tot_atraso,
      tot_adiant,
      tot_estoque,
      horizonte;
int num_lotes;
struct dados_ord *aux_ultimo;

/* Atraso ou adiantamento */

for (i=0, tot_atraso=0, tot_adiant=0; i<TOT_ORDENS; i++)
    if ((ordens+i)->fim_lam+(tp_grupo+(ordens+i)->grupo)->tp_acab[(ordens+i)-
>acab]
                                                    > (ordens+i)->prazo_entr)
        tot_atraso += ((ordens+i)->fim_lam +
              (tp_grupo+(ordens+i)->grupo)->tp_acab[(ordens+i)->acab] -
              (ordens+i)->prazo_entr)*(ordens+i)->qtde;
    else
        tot_adiant += ((ordens+i)->prazo_entr - (ordens+i)->fim_lam
         -(tp_grupo+(ordens+i)->grupo)->tp_acab[(ordens+i)->acab])*(ordens+i)-
>qtde;

/* Estoque */

for (i=0, tot_estoque = 0 ; i<TOT_ORDENS; i++)
    tot_estoque += ((ordens+i)->fim_lam - (ordens+i)->qtde *
                    (tp_faixa+(ordens+i)->faixa)->tp_unit_lam -
                    (ordens+i)->fim_desb) * (ordens+i)->qtde;

/* Set up */

for (i=0; (sol_lam+i)->inicio; i++);
num_lotes = i;

/* Horizonte */

aux_ultimo = *(ordens_seq_proc_lam+((sol_lam+num_lotes-1)->fim-
ordens_faixa_fim_desb));
horizonte = aux_ultimo->fim_lam +
            (tp_grupo+aux_ultimo->grupo)->tp_acab[aux_ultimo->acab];

if (horizon_ideal_fund>horizon_ideal_lam)
   horizonte -= horizon_ideal_fund;
else
   horizonte -= horizon_ideal_lam;

return tot_atraso * ct_atraso +
              tot_adiant * ct_adiant +
              tot_estoque * ct_estoque +
              num_lotes * ct_setup_lam +
              horizonte * ct_horizon;

}


/* Funcao adaptada de Numerical Recipes in C (vide bibliografia) */
/* Metropolis algorithm. metrop returns a boolean variable that issues a
  verdict on whether to accept a reconfiguration that leads to a change
```

```
  de in the objective function e. If de<0, metrop = 1 (true), while
  if de>0, metrop is only true wih probability exp(-de/t), where t is a
  temperature determined by the anneling schedule.                    */

int metrop(float de, float t)
{

static long gljdum=1;

if (de<0.0) return 1; /* troca */
if (de/t>10) return 0; /* nao troca */

return ran3(&gljdum) < expp(-de/t);

}



/* Gera solucao vizinha a solucao atual na fundicao. */

void Gera_vizinho_fund(struct fornada melh_sol_fund[MAX_FORNADAS],
struct fornada sol_fund[MAX_FORNADAS], int num_fornadas, int *po_forn1,
int *po_forn2)
{

register int i;
long idum=1;
float aux_ran;
struct fornada aux_troca;
int aux;

/* A geracao da vizinhaca eh baseada na ultima solucao aceita. */
for (i=0; i<num_fornadas; i++)
    *(sol_fund+i) = *(melh_sol_fund+i);

/* Sorteia duas fornadas para serem trocadas de posicao na fila */

do {
   aux_ran = ran3(&idum);
} while (aux_ran==1);
*po_forn1 = (int) (num_fornadas*aux_ran);

do {
   do {
      aux_ran = ran3(&idum);
   } while (aux_ran==1);

   *po_forn2 = (int) ((num_fornadas-1)*aux_ran);
} while ((*po_forn2-*po_forn1)>3 ||
                              (*po_forn1-*po_forn2)>4); /* nao permite sortear */
                                    /* trechos muito grandes (maximo
5). */
if (*po_forn2 >= *po_forn1) (*po_forn2)++;
else {aux=*po_forn1;
   *po_forn1=*po_forn2;
   *po_forn2=aux;
}


/* Troca de posicao. */
for (i=*po_forn1; i<(*po_forn1+*po_forn2+1)/2; i++) {
    aux_troca = *(sol_fund+i);
    *(sol_fund+i) = *(sol_fund+*po_forn2-(i-*po_forn1));
    *(sol_fund+*po_forn2-(i-*po_forn1)) = aux_troca;
}

}



/* Funcao retirada de Numerical Recipes in C (vide bibliografia). */
/* Return a uniform random deviate between 0.0 and 1.0. Set idum to any
  negative value to inicializate or reinicializate the sequence.      */

float ran3(long *idum)
```

```
{

static int inext,inextp;
static long ma[56];         /* The value 56 (range ma[1..55]) is special */
static int iff=0;           /* and should not be modified; see Knuth.    */
long mj,mk;
int i,ii,k;

if (*idum < 0 || iff == 0) {  /* Inicialization.*/
   iff=1;
   mj=MSEED-(*idum<0 ? -*idum : *idum); /* Inicializating ma using the seed */
   mj %= MBIG;                          /* seed idum and large number MSEED.*/
   ma[55]=mj;
   mk=1;

   for (i=1;i<=54;i++) {      /* Now inicializate the rest of the table,*/
       ii=(21*i) % 55;        /* in a slightly random order, with       */
       ma[ii]=mk;             /* numbers that are not especialy random. */
       mk=mj-mk;
       if (mk < MZ) mk += MBIG;
          mj=ma[ii];
   }
   for (k=1;k<=4;k++)           /* We randomize them by "warming up the */
       for (i=1;i<=55;i++) {    /* generator".                          */
           ma[i] -= ma[1+(i+30) % 55];
           if (ma[i] < MZ) ma[i] += MBIG;
       }
   inext=0;       /* Prepare indices for our first genereted number.*/
   inextp=31;     /* The constant 31 is special; see Knuth.         */
   *idum=1;
}

/* Here is where we start, except on inicialization. */

if (++inext == 56) inext=1;    /* Increment inext and inextp, wrapping       */
if (++inextp == 56) inextp=1; /*  around 56 to 1.                           */
mj=ma[inext]-ma[inextp];       /* Generate a new random number subtractively.*/
if (mj < MZ) mj += MBIG;       /* Be sure that it is in range.               */
ma[inext]=mj;                  /* Store it,                                  */
return mj*FAC;                 /* and output the derived uniform deviate.    */

}



/* Funcao para o calculo do exponencial de x, atraves da serie de Taylor
e(x)=1 + x + x2/2! + x3/3! + ...Necessaria devido a impossibilidade da
utilizacao da funcao exp() da biblioteca no compilador DJGPP */

double expp(double x)
{

double e, inc=1;
int i;

for (i=1, e=1; inc>0.0000001 || inc<-0.0000001; i++) {
    inc*=(x/i);
    e += inc;
}
return e;

}
```

ANEXO 2

DADOS DA LAMINA SA

IND	ORDEM	PARC.	GRUPOLIGA	FAIXA	ACAB.	QTDE	ENTREGA
0	8223164	1	0101W3333Y	6	BRU	20000	02/07/96
1	8223524	1	0101W3333Y	6	BRU	10000	05/07/96
2	8223220	1	0101W3333Y	6	BRU	15000	07/07/96
3	8223284	1	0101W3333Y	6	BRU	20000	10/07/96
4	7885668	1	0101W3343	2	RET	100	10/06/96
5	7888324	1	0101W3343	4	RET	100	10/06/96
6	7888804	1	0101W3343	4	RET	100	10/06/96
7	7888024	1	0101W3343	3	BRU	100	10/06/96
8	7887856	1	0101W3343	3	DES	100	10/06/96
9	7888088	1	0101W3343	4	RET	120	10/06/96
10	7885444	1	0101W3343	2	BRU	200	10/06/96
11	7885344	1	0101W3343	2	BRU	200	10/06/96
12	7887364	1	0101W3343	3	BRU	200	10/06/96
13	7887296	1	0101W3343	3	DES	200	10/06/96
14	7887176	1	0101W3343	3	BRU	200	10/06/96
15	7885872	1	0101W3343	2	DES	200	10/06/96
16	7886552	1	0101W3343	3	BRU	200	10/06/96
17	7886292	1	0101W3343	3	BRU	200	10/06/96
18	7888972	1	0101W3343	4	BRU	200	10/06/96
19	7886940	1	0101W3343	3	BRU	250	10/06/96
20	7886056	1	0101W3343	2	RET	300	10/06/96
21	7885936	1	0101W3343	2	BRU	300	10/06/96
22	7884884	1	0101W3343	1	BRU	300	10/06/96
23	7886124	1	0101W3343	3	TRE	300	10/06/96
24	7886756	1	0101W3343	3	DES	300	10/06/96
25	7888412	1	0101W3343	4	BRU	400	10/06/96
26	7884956	1	0101W3343	1	BRU	400	10/06/96
27	7889436	1	0101W3343	4	RET	3000	10/06/96
28	8228104	1	0101W3343	2	BRU	400	27/06/96
29	8213752	1	0101W3343	3	BRU	500	27/06/96
30	8213612	1	0101W3343	3	BRU	600	27/06/96
31	8227872	1	0101W3343	2	RET	600	27/06/96
32	8228884	1	0101W3343	1	DES	700	27/06/96
33	8228728	1	0101W3343	1	RET	800	27/06/96
34	8228000	1	0101W3343	2	TRE	800	27/06/96
35	8227936	1	0101W3343	2	BRU	800	27/06/96
36	8228632	1	0101W3343	1	DES	1000	27/06/96
37	8228168	1	0101W3343	2	RET	1000	27/06/96
38	8227516	1	0101W3343	3	DES	1000	27/06/96
39	8227344	1	0101W3343	3	BRU	1000	27/06/96
40	8213848	1	0101W3343	3	RET	1000	27/06/96
41	8228276	1	0101W3343	2	BRU	1000	27/06/96
42	8227408	1	0101W3343	3	DES	1500	27/06/96
43	8213492	1	0101W3343	4	RET	1500	27/06/96
44	8227192	1	0101W3343	3	RET	2000	27/06/96
45	8159964	1	0101W3343	3	TRE	350	02/07/96
46	7840300	1	0101XAM2	7	BRU	250	11/06/96
47	7566504	1	0101XAM2	4	DES	250	11/06/96
48	7510000	1	0101XAM2	5	DES	250	11/06/96
49	7575324	1	0101XAM2	1	BRU	500	11/06/96
50	7574784	1	0101XAM2	3	BRU	500	11/06/96
51	7574644	1	0101XAM2	4	TRE	750	11/06/96
52	7510136	1	0101XAM2	4	FRE	750	11/06/96
53	7511376	1	0101XAM2	2	DES	750	11/06/96
54	7575260	1	0101XAM2	2	RET	750	17/06/96
55	7840420	1	0101XAM2	7	FRE	300	24/06/96
56	6590188	1	0101XAM2	1	BRU	750	03/07/96
57	7875464	1	0101XAM2	2	DES	250	04/07/96
58	7864804	1	0101XAM2	4	DES	500	04/07/96
59	7875648	1	0101XAM2	1	BRU	500	04/07/96
60	7875788	1	0101XAM2	1	BRU	500	04/07/96
61	7864788	1	0101XAM2	3	DES	500	04/07/96
62	7864876	1	0101XAM2	3	DES	1000	04/07/96
63	7875564	1	0101XAM2	2	BRU	1000	04/07/96
64	6526668	2	0101XAM2	1	BRU	250	16/07/96
65	7807324	1	0101XAM2H	6	BRU	15000	22/06/96
66	8223748	1	0101XAM2H	6	RET	15000	05/07/96
67	7303240	1	0101XAM7D	5	TRE	450	14/06/96
68	7299780	1	0101XAM7D	5	TOR	450	14/06/96
69	7918232	1	0101XAM7D	2	BRU	50	21/06/96
70	7303376	1	0101XAM7D	4	BRU	450	25/06/96

71	7303312	1	0101XAM7D	5	TOR	450	25/06/96
72	7303516	1	0101XAM7D	3	BRU	500	25/06/96
73	7640108	1	0101XAM7D	5	RET	450	28/06/96
74	7299952	1	0101XAM7D	5	BRU	450	28/06/96
75	8106964	1	0101XAM7D	7	DES	450	30/06/96
76	8111904	1	0101XAM7D	7	RET	450	11/07/96
77	8112372	1	0101XAM7D	5	RET	450	11/07/96
78	8111732	1	0101XAM7D	7	BRU	450	11/07/96
79	8111968	1	0101XAM7D	7	RET	450	11/07/96
80	8112524	1	0101XAM7D	5	BRU	450	11/07/96
81	8112632	1	0101XAM7D	4	BRU	450	11/07/96
82	8112764	1	0101XAM7D	4	TRE	450	11/07/96
83	8113312	1	0101XAM7D	4	BRU	450	11/07/96
84	8110276	1	0101XAM7D	1	BRU	500	11/07/96
85	8113580	1	0101XAM7D	4	BRU	500	11/07/96
86	8110168	1	0101XAM7D	3	BRU	500	11/07/96
87	8110104	1	0101XAM7D	3	BRU	500	11/07/96
88	8109952	1	0101XAM7D	3	BRU	500	11/07/96
89	8112308	1	0101XAM7D	5	DES	900	11/07/96
90	8113936	1	0101XAM7D	7	BRU	450	22/07/96
91	8115176	1	0101XAM7D	4	DES	450	22/07/96
92	8114940	1	0101XAM7D	4	BRU	450	22/07/96
93	8114584	1	0101XAM7D	5	DES	450	22/07/96
94	8114196	1	0101XAM7D	7	DES	450	22/07/96
95	8114684	1	0101XAM7D	5	DES	450	22/07/96
96	8115112	1	0101XAM7D	4	FRE	450	22/07/96
97	8115232	1	0101XAM7D	3	BRU	500	22/07/96
98	8115620	1	0101XAM7D	3	DES	500	22/07/96
99	8115564	1	0101XAM7D	3	BRU	500	22/07/96
100	8114292	1	0101XAM7D	5	BRU	900	22/07/96
101	8116360	1	0101XAM7D	5	BRU	450	02/08/96
102	8116480	1	0101XAM7D	4	BRU	450	02/08/96
103	8116584	1	0101XAM7D	4	BRU	450	02/08/96
104	8116228	1	0101XAM7D	7	FRE	450	02/08/96
105	8116124	1	0101XAM7D	7	DES	450	02/08/96
106	8116444	1	0101XAM7D	5	RET	450	02/08/96
107	8116684	1	0101XAM7D	2	BRU	500	02/08/96
108	8116616	1	0101XAM7D	3	BRU	500	02/08/96
109	8116292	1	0101XAM7D	5	BRU	900	02/08/96
110	7896272	1	0101XKM42	3	BRU	1000	16/06/96
111	7896164	1	0101XKM42	3	BRU	1000	16/06/96
112	7896524	1	0101XKM42	4	BRU	2000	16/06/96
113	7896660	1	0101XKM42	4	DES	1500	16/06/96
114	8229800	1	0101XKM42	3	TRE	1200	27/06/96
115	8229872	1	0101XKM42	2	BRU	1200	27/06/96
116	8229548	1	0101XKM42	3	DES	1200	27/06/96
117	8229444	1	0101XKM42	4	BRU	500	27/06/96
118	8229936	1	0101XKM42	1	RET	500	27/06/96
119	8229732	1	0101XKM42	3	RET	600	27/06/96
120	8230056	1	0101XKM42	5	BRU	800	27/06/96
121	8229344	1	0101XKM42	5	DES	800	27/06/96
122	7275608	1	0101XKM42N	5	BRU	450	14/06/96
123	7275644	1	0101XKM42N	2	BRU	500	14/06/96
124	7275488	2	0101XKM42N	4	TRE	260	17/06/96
125	7275748	1	0101XKM42N	4	DES	450	25/06/96
126	7275716	1	0101XKM42N	4	DES	450	25/06/96
127	8109320	1	0101XKM42N	4	BRU	500	30/06/96
128	8109596	1	0101XKM42N	2	TOR	500	30/06/96
129	8109456	1	0101XKM42N	3	RET	500	30/06/96
130	7817508	1	0101XKM42N	2	BRU	1500	07/07/96
131	8110664	1	0101XKM42N	5	BRU	300	11/07/96
132	8110848	1	0101XKM42N	2	FRE	500	11/07/96
133	8110728	1	0101XKM42N	4	BRU	500	11/07/96
134	8115996	1	0101XKM42N	7	BRU	300	22/07/96
135	8116056	1	0101XKM42N	4	BRU	450	22/07/96
136	8146692	1	0101XKM42N	2	RET	2000	29/07/96
137	8116940	1	0101XKM42N	4	BRU	500	02/08/96
138	8116972	1	0101XKM42N	2	BRU	500	02/08/96
139	8117004	1	0101XKM42N	2	DES	500	02/08/96
140	8117040	1	0101XKM42N	1	RET	500	02/08/96
141	7650524	1	0103XD4Y	4	TRE	10000	15/06/96
142	7650560	3	0103XD4Y	4	BRU	2500	27/06/96
143	7650560	2	0103XD4Y	4	DES	2500	27/06/96
144	7650560	1	0103XD4Y	4	BRU	5000	27/06/96
145	7650632	1	0103XD4Y	4	BRU	10000	07/07/96

146	7726404	1	0103XD4Y	4	DES	15000	29/07/96
147	7725404	1	0103XGRYPH1	3	RET	1000	15/06/96
148	7725304	1	0103XGRYPH1	4	FRE	1000	15/06/96
149	7725368	1	0103XGRYPH1	3	BRU	1000	15/06/96
150	7725540	1	0103XGRYPH1	2	DES	1000	15/06/96
151	7725572	1	0103XGRYPH1	1	BRU	1000	15/06/96
152	7725476	1	0103XGRYPH1	2	BRU	1000	15/06/96
153	7725508	1	0103XGRYPH1	2	BRU	1000	15/06/96
154	7725628	1	0103XGRYPH1	1	RET	2000	15/06/96
155	7725336	1	0103XGRYPH1	3	BRU	1500	15/06/96
156	7570760	1	0103XN131	1	RET	500	11/06/96
157	7582484	2	0103XN131	1	BRU	500	11/06/96
158	7524528	1	0103XN131	4	BRU	500	11/06/96
159	7582300	1	0103XN131	4	DES	500	11/06/96
160	7524992	2	0103XN131	1	DES	1000	11/06/96
161	7824076	1	0103XN131	3	RET	500	24/06/96
162	7524700	1	0103XN131	2	TRE	500	24/06/96
163	7839564	1	0103XN131	1	BRU	500	24/06/96
164	7824204	1	0103XN131	2	RET	500	24/06/96
165	7824344	1	0103XN131	1	BRU	500	24/06/96
166	7824140	1	0103XN131	3	RET	500	24/06/96
167	7823972	1	0103XN131	4	TRE	1000	24/06/96
168	7825480	1	0103XN131	1	BRU	1000	24/06/96
169	7839328	1	0103XN131	2	BRU	1000	24/06/96
170	7524764	1	0103XN131	2	DES	2500	24/06/96
171	7909964	1	0103XN131	1	BRU	500	04/07/96
172	7909928	1	0103XN131	2	BRU	500	04/07/96
173	7909724	1	0103XN131	3	RET	1000	04/07/96
174	7909660	1	0103XN131	3	TOR	1000	04/07/96
175	7909628	1	0103XN131	4	RET	1000	04/07/96
176	7910016	1	0103XN131	1	RET	1500	04/07/96
177	7702252	1	0103XWD	2	BRU	300	11/06/96
178	7540412	1	0103XWD	4	BRU	500	15/06/96
179	7838056	1	0103XWD	2	FRE	500	24/06/96
180	7825736	1	0103XWD	2	TOR	500	24/06/96
181	7825908	1	0103XWD	1	BRU	500	24/06/96
182	7540756	1	0103XWD	1	RET	1000	24/06/96
183	7825804	1	0103XWD	2	BRU	1000	24/06/96
184	7838228	1	0103XWD	1	BRU	3000	24/06/96
185	7910220	1	0103XWD	2	BRU	500	04/07/96
186	7910188	1	0103XWD	3	BRU	500	04/07/96
187	7910320	1	0103XWD	1	RET	1000	04/07/96
188	7910252	1	0103XWD	2	RET	1000	04/07/96
189	7910284	1	0103XWD	1	RET	1500	04/07/96
190	7655112	1	0201W4021	4	DES	900	11/06/96
191	7655296	1	0201W4021	3	DES	950	20/06/96
192	7729848	1	0201X430G	7	BRU	5000	20/06/96
193	7794084	1	0201W4021	3	BRU	1700	24/06/96
194	7655468	1	0201W4021	3	DES	1700	01/07/96
195	7729816	1	0201X430G	7	RET	5000	01/07/96
196	7794220	1	0201W4021	2	TOR	1000	04/07/96
197	7655532	1	0201W4021	2	RET	1500	12/07/96
198	7655500	1	0201W4021	3	DES	1700	12/07/96
199	7729784	1	0201X430G	7	DES	5000	12/07/96
200	7655364	1	0201W4021	4	BRU	900	18/07/96
201	7655328	1	0201W4021	3	RET	950	23/07/96
202	7655888	1	0201W4021	2	BRU	1500	03/08/96
203	7762476	1	0201XN140D	1	BRU	560	10/06/96
204	7762284	1	0201XN140D	2	BRU	2200	17/06/96
205	8014356	1	0201XN140D	1	TOR	1500	25/06/96
206	8212016	1	0201XN140D	1	BRU	1000	07/07/96
207	8204004	1	0201XN140D	1	BRU	3000	07/07/96
208	8034288	1	0201XN140D	1	BRU	2200	13/07/96
209	7652956	1	0201XN150	5	RET	500	14/06/96
210	7652428	1	0201XN150	7	BRU	500	14/06/96
211	7395240	1	0201XN150	3	BRU	1000	19/06/96
212	8079332	1	0201XN150	1	BRU	1000	28/06/96
213	8041036	1	0201XN150	4	BRU	500	02/07/96
214	8041132	1	0201XN150	1	DES	500	02/07/96
215	8041068	1	0201XN150	3	BRU	500	02/07/96
216	8041100	1	0201XN150	2	TRE	500	02/07/96
217	8290120	1	0201XN150	1	DES	1000	04/07/96
218	8289836	1	0201XN150	4	DES	1000	06/07/96
219	8246540	1	0201XN150	7	DES	1000	07/07/96
220	8289868	1	0201XN150	3	BRU	1000	09/07/96

221	8242160	1	0201XN150	1	TOR	500	09/07/96
222	8242092	1	0201XN150	5	DES	500	09/07/96
223	8242124	1	0201XN150	4	BRU	500	09/07/96
224	8242264	1	0201XN150	1	DES	1500	09/07/96
225	8290020	1	0201XN150	2	BRU	1000	12/07/96
226	8044000	1	0201XN150	3	TOR	500	12/07/96
227	8044072	1	0201XN150	2	RET	500	12/07/96
228	8044104	1	0201XN150	1	TRE	500	12/07/96
229	8290224	1	0201XN150	1	BRU	1000	14/07/96
230	7035412	1	0202W4301	0	BRU	500	15/06/96
231	7209608	1	0202W4301	0	BRU	500	15/06/96
232	6973860	1	0202W4301	0	RET	500	15/06/96
233	6963168	1	0202W4301	0	RET	560	15/06/96
234	7213052	1	0202W4301	0	RET	1000	15/06/96
235	6787992	1	0202W4301	0	BRU	1000	15/06/96
236	6973892	1	0202W4301	0	BRU	1000	15/06/96
237	6819660	1	0202W4301	0	DES	3170	15/06/96
238	6819864	1	0202W4301	0	TOR	3500	15/06/96
239	7778524	1	0202W4301	0	TRE	500	27/06/96
240	7253992	1	0202W4301	0	BRU	500	27/06/96
241	7433632	1	0202W4301	0	RET	500	27/06/96
242	7211516	1	0202W4301	0	RET	500	27/06/96
243	7034764	1	0202W4301	0	BRU	500	27/06/96
244	6747700	1	0202W4301	0	TOR	500	27/06/96
245	7035348	1	0202W4301	0	RET	500	27/06/96
246	6747636	1	0202W4301	0	RET	500	27/06/96
247	7339836	1	0202W4301	0	BRU	500	27/06/96
248	6741860	1	0202W4301	0	BRU	500	27/06/96
249	7254380	1	0202W4301	0	DES	1000	27/06/96
250	7253360	1	0202W4301	0	DES	1000	27/06/96
251	7252936	1	0202W4301	0	DES	1000	27/06/96
252	7248812	1	0202W4301	0	BRU	1000	27/06/96
253	7213204	1	0202W4301	0	BRU	1000	27/06/96
254	7213192	1	0202W4301	0	RET	1000	27/06/96
255	7213084	1	0202W4301	0	BRU	1000	27/06/96
256	7336244	1	0202W4301	0	RET	1000	27/06/96
257	7212988	1	0202W4301	0	DES	1000	27/06/96
258	6544648	1	0202W4301	0	BRU	1000	27/06/96
259	7254436	1	0202W4301	0	RET	1000	27/06/96
260	6787852	1	0202W4301	0	BRU	1000	27/06/96
261	7348380	1	0202W4301	0	RET	1000	27/06/96
262	7253464	1	0202W4301	0	FRE	1000	27/06/96
263	6786968	1	0202W4301	0	RET	1000	27/06/96
264	6787076	1	0202W4301	0	TRE	1000	27/06/96
265	7808612	1	0202W4301	0	BRU	1000	27/06/96
266	6787108	1	0202W4301	0	BRU	1000	27/06/96
267	7253432	1	0202W4301	0	BRU	1000	27/06/96
268	6787140	1	0202W4301	0	DES	1000	27/06/96
269	6540664	1	0202W4301	0	BRU	1000	27/06/96
270	7771924	1	0202W4301	0	BRU	1500	27/06/96
271	7771144	1	0202W4301	0	BRU	1500	27/06/96
272	7772124	1	0202W4301	0	RET	1500	27/06/96
273	6819540	1	0202W4301	0	BRU	1780	27/06/96
274	7336352	1	0202W4301	0	DES	2000	27/06/96
275	7254468	1	0202W4301	0	BRU	2000	27/06/96
276	6787292	1	0202W4301	0	BRU	2000	27/06/96
277	6787324	1	0202W4301	0	DES	2000	27/06/96
278	7772228	1	0202W4301	0	BRU	2000	27/06/96
279	7253564	1	0202W4301	0	RET	2000	27/06/96
280	7253528	1	0202W4301	0	RET	2000	27/06/96
281	7253496	1	0202W4301	0	RET	2000	27/06/96
282	6819572	1	0202W4301	0	BRU	2000	27/06/96
283	7771176	1	0202W4301	0	TRE	2500	27/06/96
284	7437684	1	0202W4301	0	BRU	500	07/07/96
285	7370208	1	0202W4301	0	TOR	500	07/07/96
286	7370436	1	0202W4301	0	RET	500	07/07/96
287	8007908	1	0202W4301	0	DES	1000	07/07/96
288	7958008	1	0202W4301	0	TOR	1000	07/07/96
289	7957924	1	0202W4301	0	BRU	1000	07/07/96
290	7958504	1	0202W4301	0	TRE	1000	07/07/96
291	7958536	1	0202W4301	0	BRU	1000	07/07/96
292	8007768	1	0202W4301	0	DES	1000	07/07/96
293	7958116	1	0202W4301	0	RET	1000	07/07/96
294	8007804	1	0202W4301	0	FRE	1000	07/07/96
295	8007972	1	0202W4301	0	FRE	1000	07/07/96

296	7348316	1	0202W4301	0	DES	1000	07/07/96
297	7336828	1	0202W4301	0	BRU	1000	07/07/96
298	7365276	1	0202W4301	0	RET	1000	07/07/96
299	7348468	1	0202W4301	0	BRU	1000	07/07/96
300	7348500	1	0202W4301	0	BRU	1000	07/07/96
301	7348672	1	0202W4301	0	BRU	2000	07/07/96
302	7336116	1	0202W4301	0	BRU	2000	07/07/96
303	8007736	1	0202W4301	0	BRU	2000	07/07/96
304	7955040	1	0202W4301	0	DES	2000	07/07/96
305	7955500	1	0202W4301	0	BRU	2000	07/07/96
306	7335980	1	0202W4301	0	BRU	2000	07/07/96
307	7336044	1	0202W4301	0	BRU	2000	07/07/96
308	8008140	1	0202W4301	0	RET	3000	07/07/96
309	8008172	1	0202W4301	0	BRU	3000	07/07/96
310	7337096	1	0202W4301	0	BRU	3000	07/07/96
311	8008004	1	0202W4301	0	BRU	3000	07/07/96
312	7958180	1	0202W4301	0	RET	3000	07/07/96
313	7958568	1	0202W4301	0	RET	3000	07/07/96
314	7365372	1	0202W4301	0	RET	3000	07/07/96
315	8007940	1	0202W4301	0	BRU	4000	07/07/96
316	8008764	1	0202W4301	0	DES	6000	07/07/96
317	8163296	1	0202W4301	0	BRU	500	18/07/96
318	7613784	1	0202W4301	0	FRE	500	18/07/96
319	7613224	1	0202W4301	0	RET	500	18/07/96
320	7647596	1	0202W4301	0	DES	1000	18/07/96
321	7647524	1	0202W4301	0	FRE	1000	18/07/96
322	7647488	1	0202W4301	0	BRU	1000	18/07/96
323	7647456	1	0202W4301	0	BRU	1000	18/07/96
324	8163176	1	0202W4301	0	BRU	1000	18/07/96
325	7594944	1	0202W4301	0	RET	1500	18/07/96
326	7646776	1	0202W4301	0	BRU	2000	18/07/96
327	7646964	1	0202W4301	0	DES	2000	18/07/96
328	7647608	1	0202W4301	0	FRE	2000	18/07/96
329	7597044	1	0202W4301	0	BRU	2000	18/07/96
330	5909076	1	0202W4301D	0	RET	500	27/06/96
331	7387724	1	0202W4301D	0	RET	500	27/06/96
332	7388416	1	0202W4301D	0	TOR	500	27/06/96
333	7388092	1	0202W4301D	0	BRU	1000	07/07/96
334	7387860	1	0202W4301D	0	BRU	6000	07/07/96
335	8221960	1	0202W4301D	0	BRU	1000	18/07/96
336	8222184	1	0202W4301D	0	BRU	1000	18/07/96
337	8222152	1	0202W4301D	0	BRU	1500	18/07/96
338	8222044	1	0202W4301D	0	BRU	2500	18/07/96
339	6679628	1	0202W4301I	0	BRU	1000	15/06/96
340	6681516	1	0202W4301I	0	BRU	500	21/06/96
341	6678980	1	0202W4301I	0	BRU	500	27/06/96
342	6263572	2	0202W4301I	0	DES	500	27/06/96
343	6678452	1	0202W4301I	0	DES	500	27/06/96
344	6678624	1	0202W4301I	0	DES	500	27/06/96
345	6678808	1	0202W4301I	0	DES	500	27/06/96
346	6679368	1	0202W4301I	0	BRU	1000	27/06/96
347	6678688	1	0202W4301I	0	DES	1000	27/06/96
348	6264164	1	0202W4301I	0	BRU	1000	27/06/96
349	6258768	1	0202W4301I	0	BRU	1000	27/06/96
350	6679508	1	0202W4301I	0	BRU	1000	27/06/96
351	6263692	1	0202W4301I	0	RET	1500	27/06/96
352	6571844	1	0202W4436Y	0	RET	500	15/06/96
353	6901544	1	0202W4436Y	0	BRU	1000	15/06/96
354	7202684	1	0202W4436Y	0	RET	1000	15/06/96
355	7202556	1	0202W4436Y	0	BRU	1000	15/06/96
356	7153552	1	0202W4436Y	0	TOR	1000	15/06/96
357	6571284	1	0202W4436Y	0	DES	1000	15/06/96
358	6572008	1	0202W4436Y	0	TOR	1000	15/06/96
359	6901204	1	0202W4436Y	0	BRU	1000	15/06/96
360	7308684	1	0202W4436Y	0	DES	500	27/06/96
361	7014132	1	0202W4436Y	0	BRU	500	27/06/96
362	6901188	1	0202W4436Y	0	DES	500	27/06/96
363	6901276	1	0202W4436Y	0	BRU	1000	27/06/96
364	7014164	1	0202W4436Y	0	TOR	1000	27/06/96
365	6571744	1	0202W4436Y	0	BRU	1000	27/06/96
366	7937492	1	0202W4436Y	0	BRU	1000	07/07/96
367	7936836	1	0202W4436Y	0	DES	1000	07/07/96
368	7936748	1	0202W4436Y	0	TOR	1000	07/07/96
369	7911428	1	0202W4436Y	0	FRE	3000	07/07/96
370	6965784	2	0202W4541	0	BRU	370	15/06/96

371	6744388	1	0202W4541	0	DES	400	15/06/96
372	6965680	1	0202W4541	0	DES	1000	15/06/96
373	7356392	1	0202W4541	0	RET	1000	15/06/96
374	6821628	1	0202W4541	0	BRU	1000	15/06/96
375	6821660	1	0202W4541	0	BRU	1000	15/06/96
376	7356544	1	0202W4541	0	BRU	1700	15/06/96
377	6965644	1	0202W4541	0	DES	500	27/06/96
378	6420284	1	0202W4541	0	BRU	500	27/06/96
379	6420248	1	0202W4541	0	BRU	1000	27/06/96
380	6965752	1	0202W4541	0	BRU	1000	27/06/96
381	7376592	1	0202W4541	0	DES	500	07/07/96
382	7376656	1	0202W4541	0	DES	500	07/07/96
383	7377200	1	0202W4541	0	TOR	1000	07/07/96
384	7433276	1	0202W4541	0	BRU	1000	07/07/96
385	7377152	1	0202W4541	0	BRU	1000	07/07/96
386	7365728	1	0202W4541	0	RET	1000	07/07/96
387	7433204	1	0202W4541	0	TOR	2000	07/07/96
388	6838696	1	0202W4571	0	BRU	500	15/06/96
389	6975036	1	0202W4571	0	FRE	500	15/06/96
390	6253764	2	0202W4571	0	BRU	500	27/06/96
391	6559244	1	0202W4571	0	BRU	500	27/06/96
392	7792808	1	0202W4571	0	TRE	500	27/06/96
393	7793012	1	0202W4571	0	BRU	500	27/06/96
394	6822964	1	0202W4571	0	RET	500	27/06/96
395	7371448	1	0202W4571	0	BRU	1000	27/06/96
396	7374364	1	0202W4571	0	TOR	1000	27/06/96
397	6822932	1	0202W4571	0	TRE	1000	27/06/96
398	6443284	1	0202W4571	0	FRE	1000	27/06/96
399	7357000	1	0202W4571	0	BRU	1000	27/06/96
400	6443320	1	0202W4571	0	BRU	1000	27/06/96
401	7214196	1	0202W4571	0	DES	500	07/07/96
402	7374096	1	0202W4571	0	BRU	500	07/07/96
403	7374300	1	0202W4571	0	RET	1000	07/07/96
404	7374592	1	0202W4571	0	RET	1000	07/07/96
405	7374520	1	0202W4571	0	BRU	4000	07/07/96
406	7996704	1	0202X302	3	BRU	5000	14/06/96
407	7996532	1	0202X302	3	FRE	5000	14/06/96
408	8246476	1	0202X302	2	DES	2000	01/07/96
409	8246248	1	0202X302	4	BRU	2000	01/07/96
410	8246284	1	0202X302	4	BRU	2000	01/07/96
411	8246404	1	0202X302	2	RET	2000	01/07/96
412	8246356	1	0202X302	3	DES	4000	01/07/96
413	8246388	1	0202X302	3	BRU	4000	01/07/96
414	8229376	1	0202X302J	7	FRE	12000	27/06/96
415	7806612	1	0202X303	3	DES	500	11/06/96
416	7813160	1	0202X303	3	TOR	500	15/06/96
417	7650664	1	0202X303	5	BRU	500	15/06/96
418	7623812	1	0202X303	4	BRU	500	15/06/96
419	7813588	1	0202X303	2	RET	1000	21/06/96
420	8027064	1	0202X303	4	BRU	500	22/06/96
421	8027128	1	0202X303	7	TRE	500	22/06/96
422	8124884	1	0202X303	1	BRU	2000	25/06/96
423	8045376	1	0202X303	5	BRU	500	27/06/96
424	8045444	1	0202X303	2	BRU	500	27/06/96
425	7934188	1	0202X303	2	RET	500	27/06/96
426	8125600	1	0202X303	3	RET	1000	30/06/96
427	8125480	1	0202X303	1	TOR	1000	30/06/96
428	8046444	1	0202X303	5	BRU	500	04/07/96
429	8245624	1	0202X303	3	BRU	500	04/07/96
430	8245536	1	0202X303	4	BRU	500	04/07/96
431	8245504	1	0202X303	5	DES	500	04/07/96
432	8245688	1	0202X303	2	TRE	1000	04/07/96
433	8245656	1	0202X303	3	BRU	2000	04/07/96
434	8186544	1	0202X303	7	FRE	500	04/07/96
435	8047856	1	0202X303	4	DES	500	05/07/96
436	8041488	1	0202X303	7	BRU	500	07/07/96
437	8041284	1	0202X303	4	TRE	500	09/07/96
438	8041392	1	0202X303	4	BRU	500	09/07/96
439	8041424	1	0202X303	3	BRU	500	09/07/96
440	8041596	1	0202X303	4	RET	500	09/07/96
441	8041456	1	0202X303	7	BRU	500	09/07/96
442	8041524	1	0202X303	7	TRE	500	13/07/96
443	8020428	2	0202X303	4	BRU	1000	14/07/96
444	8044664	1	0202X303	4	TOR	500	14/07/96
445	8044728	1	0202X303	3	DES	500	18/07/96

446	7997708	1	0202X304	3	RET	5000	14/06/96
447	7979620	1	0202X304	2	BRU	1000	16/06/96
448	7978804	1	0202X304	2	DES	1000	16/06/96
449	7979924	1	0202X304	1	BRU	1500	16/06/96
450	7983216	1	0202X304	2	BRU	1000	17/06/96
451	7983388	1	0202X304	1	BRU	1000	17/06/96
452	7983248	1	0202X304	2	BRU	2000	17/06/96
453	7385348	1	0202X304	3	BRU	1000	19/06/96
454	7385380	1	0202X304	3	DES	1000	19/06/96
455	7385316	1	0202X304	3	DES	1000	19/06/96
456	7385208	1	0202X304	3	BRU	2000	20/06/96
457	6785728	1	0202X304	7	RET	460	02/07/96
458	6789172	1	0202X304	4	BRU	500	02/07/96
459	6786052	1	0202X304	4	BRU	500	02/07/96
460	6785868	1	0202X304	5	BRU	500	02/07/96
461	6785900	1	0202X304	5	BRU	500	13/07/96
462	7628728	1	0202X304	0	BRU	2500	14/07/96
463	7628764	1	0202X304	0	TRE	3000	14/07/96
464	7628956	1	0202X304	0	DES	1000	18/07/96
465	7629580	1	0202X304	0	BRU	1000	18/07/96
466	8270600	1	0202X304	2	BRU	500	18/07/96
467	8270348	1	0202X304	4	BRU	500	18/07/96
468	6569232	1	0202X304E	0	TRE	340	27/06/96
469	6012788	1	0202X304E	0	RET	500	27/06/96
470	6550880	1	0202X304E	0	BRU	740	27/06/96
471	6569176	1	0202X304E	0	DES	1000	27/06/96
472	6898880	1	0202X304E	0	DES	500	07/07/96
473	7379136	1	0202X304E	0	BRU	500	07/07/96
474	6899836	1	0202X304E	0	TRE	1000	07/07/96
475	7935460	1	0202X304E	0	BRU	1000	07/07/96
476	7934900	1	0202X304E	0	BRU	1000	07/07/96
477	7931660	1	0202X304E	0	TRE	1000	07/07/96
478	6899512	1	0202X304E	0	FRE	1000	07/07/96
479	6898984	1	0202X304E	0	BRU	1000	07/07/96
480	7202124	1	0202X304E	0	DES	1000	07/07/96
481	7013356	1	0202X304E	0	BRU	1000	07/07/96
482	7816300	1	0202X304V	1	DES	500	11/06/96
483	7816688	1	0202X304V	1	BRU	500	12/06/96
484	7630444	1	0202X304V	0	RET	500	12/06/96
485	7630584	1	0202X304V	0	DES	500	12/06/96
486	7653344	1	0202X304V	3	BRU	500	14/06/96
487	7653204	1	0202X304V	4	BRU	500	14/06/96
488	7624012	1	0202X304V	4	BRU	1000	14/06/96
489	7607080	1	0202X304V	0	DES	500	14/06/96
490	7605384	1	0202X304V	0	BRU	1000	14/06/96
491	7605556	1	0202X304V	0	BRU	1000	14/06/96
492	7653312	1	0202X304V	4	BRU	500	15/06/96
493	7653192	1	0202X304V	5	RET	500	15/06/96
494	7624080	1	0202X304V	3	BRU	1000	15/06/96
495	7624044	1	0202X304V	3	RET	1000	16/06/96
496	7695976	1	0202X304V	7	BRU	500	16/06/96
497	7684008	1	0202X304V	0	BRU	500	16/06/96
498	7684148	1	0202X304V	0	RET	500	17/06/96
499	7624184	1	0202X304V	1	BRU	500	20/06/96
500	7624152	1	0202X304V	2	BRU	1000	21/06/96
501	7624216	1	0202X304V	1	BRU	1000	21/06/96
502	7607336	1	0202X304V	0	BRU	500	21/06/96
503	7656024	1	0202X304V	5	BRU	600	21/06/96
504	7812856	1	0202X304V	1	BRU	500	21/06/96
505	8039416	1	0202X304V	3	BRU	500	22/06/96
506	8039244	1	0202X304V	5	BRU	500	22/06/96
507	7936220	1	0202X304V	0	DES	1000	24/06/96
508	7817184	1	0202X304V	1	BRU	500	24/06/96
509	7817044	1	0202X304V	1	TRE	500	24/06/96
510	7816980	1	0202X304V	1	RET	500	24/06/96
511	8039740	1	0202X304V	3	BRU	500	25/06/96
512	8038436	1	0202X304V	2	RET	1000	27/06/96
513	8039772	1	0202X304V	1	RET	500	27/06/96
514	8045052	1	0202X304V	2	RET	1000	27/06/96
515	8045084	1	0202X304V	1	BRU	2000	27/06/96
516	7836816	1	0202X304V	4	RET	500	28/06/96
517	7837204	1	0202X304V	4	BRU	500	28/06/96
518	7942720	1	0202X304V	0	BRU	500	28/06/96
519	7921544	1	0202X304V	7	BRU	2000	28/06/96
520	8072284	1	0202X304V	3	RET	1000	28/06/96

521	8072048	1	0202X304V	5	DES	1000	28/06/96
522	8072392	1	0202X304V	2	BRU	1000	28/06/96
523	8070576	1	0202X304V	7	RET	1000	29/06/96
524	8071956	1	0202X304V	7	BRU	1000	29/06/96
525	8072440	1	0202X304V	1	BRU	1000	29/06/96
526	8072220	1	0202X304V	4	DES	1500	29/06/96
527	8079828	1	0202X304V	2	DES	1000	29/06/96
528	8079724	1	0202X304V	3	FRE	1000	29/06/96
529	8047532	1	0202X304V	3	RET	1000	30/06/96
530	8079796	1	0202X304V	2	RET	1000	02/07/96
531	8079688	1	0202X304V	4	BRU	1000	02/07/96
532	8079860	1	0202X304V	1	TRE	1000	02/07/96
533	8247132	1	0202X304V	5	DES	1000	04/07/96
534	8247036	1	0202X304V	2	BRU	2000	04/07/96
535	8247164	1	0202X304V	4	DES	2000	04/07/96
536	8246964	1	0202X304V	3	BRU	2000	04/07/96
537	7656684	1	0202X304V	4	TRE	600	04/07/96
538	8270888	1	0202X304V	2	BRU	2000	04/07/96
539	8270564	1	0202X304V	4	BRU	2000	04/07/96
540	8041608	1	0202X304V	5	RET	500	07/07/96
541	8041716	1	0202X304V	3	BRU	500	07/07/96
542	8041748	1	0202X304V	3	RET	500	07/07/96
543	8041780	1	0202X304V	3	DES	500	07/07/96
544	8041644	1	0202X304V	4	BRU	500	07/07/96
545	8271212	1	0202X304V	1	RET	1000	07/07/96
546	8042136	1	0202X304V	5	RET	500	07/07/96
547	8042104	1	0202X304V	7	BRU	500	07/07/96
548	8041844	1	0202X304V	1	DES	500	07/07/96
549	8046480	1	0202X304V	3	DES	1000	07/07/96
550	8270672	1	0202X304V	3	BRU	2000	07/07/96
551	8046552	1	0202X304V	3	TRE	1000	09/07/96
552	8270736	1	0202X304V	3	BRU	2000	09/07/96
553	8025144	1	0202X304V	4	BRU	500	09/07/96
554	7927976	1	0202X304V	0	RET	500	09/07/96
555	7931080	1	0202X304V	0	BRU	500	09/07/96
556	7943116	1	0202X304V	0	DES	500	09/07/96
557	7928244	1	0202X304V	0	RET	500	09/07/96
558	8215348	1	0202X304V	2	RET	500	09/07/96
559	8214852	1	0202X304V	7	RET	500	09/07/96
560	8215804	1	0202X304V	1	RET	500	09/07/96
561	8215024	1	0202X304V	7	BRU	1000	09/07/96
562	8217480	1	0202X304V	7	BRU	1000	09/07/96
563	8218232	1	0202X304V	2	DES	1000	09/07/96
564	8218028	1	0202X304V	3	BRU	1000	09/07/96
565	8218364	1	0202X304V	1	RET	1000	09/07/96
566	8217804	1	0202X304V	5	BRU	1500	09/07/96
567	8215088	1	0202X304V	5	DES	1500	09/07/96
568	8215208	1	0202X304V	3	BRU	1500	09/07/96
569	8217908	1	0202X304V	4	BRU	2000	09/07/96
570	7871188	1	0202X304V	1	BRU	500	09/07/96
571	7870984	1	0202X304V	1	DES	500	09/07/96
572	7870760	1	0202X304V	1	TRE	500	09/07/96
573	8046648	1	0202X304V	1	BRU	500	11/07/96
574	8046584	1	0202X304V	2	BRU	1000	11/07/96
575	8270824	1	0202X304V	2	RET	1000	11/07/96
576	8271160	1	0202X304V	1	BRU	1000	11/07/96
577	8046788	1	0202X304V	1	BRU	1000	12/07/96
578	8046616	1	0202X304V	1	BRU	1000	12/07/96
579	8270468	1	0202X304V	5	DES	2000	12/07/96
580	8247284	1	0202X304V	0	BRU	1000	14/07/96
581	7871544	1	0202X304V	1	BRU	500	14/07/96
582	7871308	1	0202X304V	1	BRU	500	14/07/96
583	7871372	1	0202X304V	1	RET	500	14/07/96
584	8247220	1	0202X304V	0	BRU	1000	17/07/96
585	8270500	1	0202X304V	3	BRU	1200	18/07/96
586	8271124	1	0202X304V	1	BRU	1200	18/07/96
587	8270244	1	0202X304V	0	BRU	2000	18/07/96
588	8248476	1	0202X304V	4	DES	500	19/07/96
589	8024876	1	0202X304V	4	BRU	500	20/07/96
590	8172068	1	0202X304V	0	BRU	500	20/07/96
591	8172036	1	0202X304V	0	DES	500	20/07/96
592	7656056	1	0202X304V	5	BRU	600	27/07/96
593	8247456	1	0202X304V	0	BRU	1000	29/07/96
594	8025232	1	0202X304V	4	BRU	500	31/07/96
595	7871868	1	0202X304V	1	BRU	500	31/07/96

596	7871632	1	0202X304V	1	BRU	500	31/07/96
597	7871728	1	0202X304V	1	BRU	500	03/08/96
598	8228560	1	0202X308LA	7	DES	24000	27/06/96
599	8228988	1	0202X308LQ	7	FRE	12000	27/06/96
600	8228460	1	0202X309LJ	7	BRU	12000	27/06/96
601	7980756	1	0202X310	1	BRU	500	14/06/96
602	7623952	1	0202X310	4	BRU	500	25/06/96
603	7623880	1	0202X310	5	TOR	500	25/06/96
604	7934608	1	0202X310	4	BRU	500	27/06/96
605	8045408	1	0202X310	7	BRU	500	27/06/96
606	8078840	1	0202X310	4	BRU	1000	28/06/96
607	8045516	1	0202X310	3	DES	500	29/06/96
608	8045580	1	0202X310	1	BRU	500	29/06/96
609	8045548	1	0202X310	2	RET	500	29/06/96
610	8042204	1	0202X310	3	BRU	500	07/07/96
611	8042168	1	0202X310	5	BRU	500	07/07/96
612	8042276	1	0202X310	3	BRU	500	07/07/96
613	8042308	1	0202X310	2	BRU	500	07/07/96
614	8241480	1	0202X310	5	BRU	1000	09/07/96
615	8241584	1	0202X310	5	BRU	1000	09/07/96
616	8241600	1	0202X310	4	BRU	1000	09/07/96
617	8241672	1	0202X310	3	BRU	1500	09/07/96
618	6795692	1	0202X310	7	BRU	500	13/07/96
619	6795864	1	0202X310	7	BRU	500	13/07/96
620	6795628	1	0202X310	7	DES	500	13/07/96
621	6795724	1	0202X310	7	BRU	1000	13/07/96
622	6708252	1	0202X316C	0	DES	500	15/06/96
623	6708188	1	0202X316C	0	BRU	500	15/06/96
624	7245284	1	0202X316C	0	RET	1000	15/06/96
625	7798820	1	0202X316C	0	BRU	500	27/06/96
626	7798764	1	0202X316C	0	BRU	500	27/06/96
627	7954196	1	0202X316C	0	FRE	500	07/07/96
628	7954292	1	0202X316C	0	TOR	500	07/07/96
629	7954480	1	0202X316C	0	BRU	500	07/07/96
630	7954360	1	0202X316C	0	RET	1000	07/07/96
631	7245844	1	0202X316C	0	RET	1000	07/07/96
632	7954324	1	0202X316C	0	FRE	1000	07/07/96
633	7954552	1	0202X316C	0	DES	1000	07/07/96
634	7245420	1	0202X316C	0	BRU	1000	07/07/96
635	7245216	1	0202X316C	0	DES	1000	07/07/96
636	7245248	1	0202X316C	0	BRU	1000	07/07/96
637	7245452	1	0202X316C	0	TRE	1000	07/07/96
638	7649320	1	0202X316C	0	BRU	1000	18/07/96
639	7645892	1	0202X316C	0	RET	1000	18/07/96
640	7649252	1	0202X316C	0	RET	1000	18/07/96
641	7624476	1	0202X316V	4	BRU	500	14/06/96
642	6920860	1	0202X316V	3	BRU	1000	15/06/96
643	6920924	1	0202X316V	3	BRU	2000	15/06/96
644	7624540	1	0202X316V	1	DES	500	25/06/96
645	7624508	1	0202X316V	2	BRU	500	25/06/96
646	7551228	1	0202X316V	1	DES	1000	25/06/96
647	7934644	1	0202X316V	3	RET	500	27/06/96
648	7934780	1	0202X316V	1	BRU	500	27/06/96
649	8040152	1	0202X316V	4	DES	500	27/06/96
650	7915600	1	0202X316V	2	BRU	2000	27/06/96
651	8073492	1	0202X316V	7	BRU	500	28/06/96
652	8073816	1	0202X316V	2	FRE	500	28/06/96
653	8040508	1	0202X316V	1	BRU	500	29/06/96
654	8047468	1	0202X316V	2	BRU	500	29/06/96
655	8045636	1	0202X316V	5	DES	500	29/06/96
656	8045668	1	0202X316V	4	TRE	500	29/06/96
657	8047328	1	0202X316V	3	RET	500	29/06/96
658	8045764	1	0202X316V	3	BRU	500	29/06/96
659	8045732	1	0202X316V	3	BRU	500	29/06/96
660	8047364	1	0202X316V	4	BRU	500	29/06/96
661	8040388	1	0202X316V	2	BRU	500	29/06/96
662	8040284	1	0202X316V	3	BRU	500	29/06/96
663	8045700	1	0202X316V	3	DES	500	29/06/96
664	8047436	1	0202X316V	3	BRU	1000	29/06/96
665	8047684	1	0202X316V	2	BRU	2000	29/06/96
666	8246508	1	0202X316V	5	BRU	3000	06/07/96
667	8043408	1	0202X316V	1	TRE	500	09/07/96
668	8043344	1	0202X316V	1	BRU	500	09/07/96
669	8043020	1	0202X316V	4	RET	500	09/07/96
670	8043312	1	0202X316V	2	RET	500	09/07/96

671	8043204	1	0202X316V	2	BRU	500	09/07/96
672	8042988	1	0202X316V	4	BRU	500	09/07/96
673	8043084	1	0202X316V	3	RET	500	09/07/96
674	8043120	1	0202X316V	3	BRU	500	09/07/96
675	8043192	1	0202X316V	3	BRU	500	09/07/96
676	8042848	1	0202X316V	5	TOR	500	09/07/96
677	8281684	1	0202X316V	4	DES	1000	09/07/96
678	8281720	1	0202X316V	3	DES	1000	12/07/96
679	8281792	1	0202X316V	1	BRU	1000	14/07/96
680	7687048	1	0306XYT31X	2	DES	6010	10/06/96
681	8006852	1	0306XYT31X	2	RET	4000	03/07/96

ORDENS

DATA PARA CALCULO DO PRAZO: 10/06/96

0	0	6	0	2000	528	22
0	0	6	0	1000	600	25
0	0	6	0	1500	648	27
0	0	6	0	2000	720	30
0	1	2	3	10	0	0
0	1	4	3	10	0	0
0	1	4	3	10	0	0
0	1	3	0	10	0	0
0	1	3	2	10	0	0
0	1	4	3	12	0	0
0	1	2	0	20	0	0
0	1	2	0	20	0	0
0	1	3	0	20	0	0
0	1	3	2	20	0	0
0	1	3	0	20	0	0
0	1	2	2	20	0	0
0	1	3	0	20	0	0
0	1	3	0	20	0	0
0	1	4	0	20	0	0
0	1	3	0	25	0	0
0	1	2	3	30	0	0
0	1	2	0	30	0	0
0	1	1	0	30	0	0
0	1	3	1	30	0	0
0	1	3	2	30	0	0
0	1	4	0	40	0	0
0	1	1	0	40	0	0
0	1	4	3	300	0	0
0	1	2	0	40	408	17
0	1	3	0	50	408	17
0	1	3	0	60	408	17
0	1	2	3	60	408	17
0	1	1	2	70	408	17
0	1	1	3	80	408	17
0	1	2	1	80	408	17
0	1	2	0	80	408	17
0	1	1	2	100	408	17
0	1	2	3	100	408	17
0	1	3	2	100	408	17
0	1	3	0	100	408	17
0	1	3	3	100	408	17
0	1	2	0	100	408	17
0	1	3	2	150	408	17
0	1	4	3	150	408	17
0	1	3	3	200	408	17
0	1	3	1	35	528	22
0	2	7	0	25	24	1
0	2	4	2	25	24	1
0	2	5	2	25	24	1
0	2	1	0	50	24	1
0	2	3	0	50	24	1
0	2	4	1	75	24	1
0	2	4	5	75	24	1
0	2	2	2	75	24	1
0	2	2	3	75	168	7
0	2	7	5	30	336	14
0	2	1	0	75	552	23
0	2	2	2	25	576	24
0	2	4	2	50	576	24

0	2	1	0	50	576	24
0	2	1	0	50	576	24
0	2	3	2	50	576	24
0	2	3	2	100	576	24
0	2	2	0	100	576	24
0	2	1	0	25	864	36
0	3	6	0	1500	288	12
0	3	6	3	1500	600	25
0	4	5	1	45	96	4
0	4	5	4	45	96	4
0	4	2	0	5	264	11
0	4	4	0	45	360	15
0	4	5	4	45	360	15
0	4	3	0	50	360	15
0	4	5	3	45	432	18
0	4	5	0	45	432	18
0	4	7	2	45	480	20
0	4	7	3	45	744	31

TP_GRUPO

8	0.001111	40	72	48	64	56	56
8	0.001111	40	56	48	56	56	56
9	0.001000	32	56	56	48	56	48
9	0.000952	24	56	40	48	56	56
8	0.001020	16	72	32	48	0	0

TP_FAIXA

15	0.006060
4	0.005050
9	0.004328
10	0.004567
7	0.006537
8	0.005128
5	0.006803
5	0.005128

ANEXO 3

Melhor solução encontrada: custo de R$ 453.770,00. São mostradas as fornadas, seus instantes de término de fundição e as ordens que pertencem a elas, identificadas por sua posição no arquivo. Os lotes de laminação também são apresentados, com o instante de término de laminação acompanhando cada ordem.

```
fornada 0/8: 65
fornada 1/16: 448 447 446 451 450 449 454 455 453 457 458 459 460 452
fornada 2/40: 680 681
fornada 3/48: 141 143 142 144
fornada 4/55: 111 110 113 118 117 112 119 121 120 114 116 115
fornada 5/64: 51 52 53 49 50 47 48 46 54 55 57 64 58 61 60 59 56 62 63
fornada 6/56: 407 406 411 408
fornada 7/63: 27 23 20 25 26 24 22 21 19 13 15 18 17 12 14 16 10 11 9 5 6 4 8 7
28 45 29 31 30 32 34 33 35 40 37 36 38 39 41 43 42 44
fornada 8/80: 599
fornada 9/70: 148 147 150 151 149 153 152 155 154
fornada 10/89: 339 340 344 345 343 342 341 347 349 350 346 348 351
fornada 11/99: 414
fornada 12/109: 203 204 205 206 208 207
fornada 13/116: 160 156 159 158 157 166 161 162 164 163 165 171 172 167 168 169
175 174 173 170 176
fornada 14/124: 232 233 231 230 234 236 235 239 244 241 242 245 246 248 247 240
243 238 237 285 286 284 262 264 259 261 263 256
fornada 15/132: 177 178 180 179 181 186 185 182 183 187 188 189 184
fornada 16/126: 600
fornada 17/153: 484 482 485 483 489 487 486 493 492 498 488 491 490 497 496 494
495 499 502 504 506 505 503 509 510 508 511 513 516 517 518 501 500 507 540 542
543 546 544 541 548
fornada 18/161: 410 409 412 413
fornada 19/169: 664 677 678 650 679 665 666
fornada 20/179: 371 370 373 372 375 374 376 377 378 381 382 379 380 383 386 385
384 387
fornada 21/188: 598
fornada 22/197: 352 356 358 354 357 359 353 355 362 360 361 364 365 363 368 367
366 369
fornada 23/206: 641 642 644 645 647 649 648 652 643 656 657 651 655 663 660 653
659 662 658 661 654 646 676 667 669 673 670 672 668 674 675 671
fornada 24/214: 190 191 193 196 194 200 201 192 197 198
fornada 25/223: 202 195 199
fornada 26/233: 461 456 467 466 464 465 462 463
fornada 27/241: 468 469 470 472 473 471 477 478 474 480 481 479 475 476
fornada 28/276: 209 210 211 214 216 213 215 222 221 223 228 226 227 212 217 218
219 220 225 229 224
fornada 29/285: 66
fornada 30/293: 567 566 568 535 586 585 539 536 538 534
fornada 31/303: 389 388 392 394 391 393 390 401 402 398 396 397 395 399 400 404
403 405
fornada 32/312: 601 603 602 604 605 609 607 608 611 612 610 613 620 618 619 606
616 614 615 621 617
fornada 33/321: 593 550 552 569 579 587
fornada 34/328: 122 123 124 126 125 129 131 128 127 134 132 133 135 140 139 137
138 130 136
fornada 35/338: 332 331 330 333 336 335 337 338 334
fornada 36/321: 415 416 418 417 421 420 425 423 424 419 434 431 430 428 429 435
436 437 440 438 441 439 442 444 427 426 445 432 422 443 433
fornada 37/328: 0
fornada 38/335: 1 2
fornada 39/345: 537 547 554 557 559 560 558 572 556 571 553 555 570 512 514 573
583 520 521 528 582 581 523 522 527 524 525 529 588 591 532 589 590 530 531
fornada 40/354: 622 623 624 626 625 628 627 629 632 637 630 631 635 633 636 634
639 640 638
fornada 41/362: 275 282 278 283 321 320 324 323 322 304 303 305 301 302
fornada 42/372: 298 293 287 292 296 291 289 299 297 300 273 280 279 281 277 274
276
fornada 43/381: 533 545 594 596 595 549 526 551 597 565 563 592 562 561 564 575
576 574 515 578 577 580 519 584
fornada 44/390: 313 310 309 311 315 316
fornada 45/399: 254 249 268 257 251 250 267 269 266 255 260 258 265 252 253 318
319 317 272 271 270 294 290 288 295
```

fornada 46/406: 67 68 69 71 70 72 73 74 75 82 79 76 77 83 81 80 78 85 84 88 87 86
96 91 93 94 95 92 90 98 99 97 106 104 105 103 102 101 108 107 89 100 109
fornada 47/414: 145 146
fornada 48/422: 3
fornada 49/433: 307 306 325 328 327 329 326 308 312 314

lote 0: 65/51
lote 1: 111/61 110/62 407/64 23/64 24/64 13/64 19/64 14/64 16/65 17/65 12/65 8/65
7/65 50/65 45/65 455/66 454/66 29/66 446/69 453/69 119/69 30/70 61/70 406/72
147/73 149/73 155/74 40/74 38/75 39/75 114/76 116/76 42/77 62/77 44/78
lote 2: 27/87 51/88 52/88 25/88 9/89 18/89 5/89 6/89 47/89 148/90 113/91 117/91
112/92 458/93 459/93 58/93 43/94 141/101 143/102 142/104 144/107
lote 3: 680/119 20/119 53/119 21/119 15/120 11/120 10/120 4/120 54/120 150/121
153/121 152/121 448/122 447/122 57/122 162/123 164/123 28/123 450/123 31/124
34/124 172/124 35/125 452/125 169/126 204/127 37/127 41/128 115/128 63/129
170/130 411/131 408/131 681/133 177/133 180/134 179/134 185/134 183/134 188/135
lote 4: 160/139 203/140 26/140 156/140 22/140 157/140 49/141 151/141 154/142
163/143 165/143 181/143 451/144 118/144 449/145 64/145 32/145 33/145 171/146
60/146 59/146 182/147 168/147 36/148 56/148 205/149 187/149 206/150 176/151
189/151 184/153 482/153 483/153 499/154 504/154 509/154 510/154 508/155 513/155
501/155 548/156 208/157 207/158
lote 5: 484/174 485/174 489/174 232/174 233/175 231/175 230/175 498/176 497/176
491/177 490/177 234/178 235/178 236/179 371/179 370/180 373/180 372/181 375/181
374/182 339/183 502/183 340/183 239/183 244/184 245/184 246/184 241/185 242/185
344/185 342/186 343/186 377/186 345/186 376/188 243/188 248/188 240/188 247/189
378/189 341/189 518/190 238/192 237/194 507/194 285/195 286/195 381/195 382/196
284/196 262/196 264/197 352/197 358/198 356/199 354/199 357/200 353/200 359/201
355/202 362/202 360/202 361/202 364/203 263/204 261/204 259/205 256/206 347/206
346/207 379/207 365/208 350/209 348/209 380/210 349/210 363/211 351/212 368/212
383/213 386/214 367/214 385/215 366/216 384/216 387/217 369/219
lote 6: 46/224 496/225 55/225 651/225 457/225 192/228 195/230 199/233 599/239
414/245 600/251 598/264
lote 7: 48/272 493/272 506/272 503/273 655/273 460/273 121/274 120/274 540/274
546/274 676/275 461/275 666/276
lote 8: 486/287 642/287 494/288 495/288 505/288 166/289 161/289 511/289 191/289
647/290 211/290 657/290 663/291 643/291 662/292 658/292 659/292 215/292 186/293
542/293 543/293 541/293 673/294 675/294 674/294 226/294 664/295 456/296 193/296
173/297 174/297 220/298 678/298 194/299 201/299 568/300 585/301 536/302 198/302
412/304 413/306
lote 9: 190/314 159/314 158/314 487/315 641/315 178/315 492/316 488/316 602/317
649/317 604/317 516/317 656/318 517/318 660/318 213/319 167/319 544/320 669/320
223/320 672/321 418/321 420/321 430/322 435/322 437/322 440/323 438/323 444/323
606/324 467/324 175/325 218/326 677/326 616/327 443/328 200/328 410/330 124/330
126/330 125/330 127/331 133/331 135/331 137/332 409/333 535/334 539/335 569/337
lote 10: 123/346 645/346 425/346 424/347 652/347 609/347 419/348 128/348 654/348
661/348 216/348 500/349 613/349 558/349 670/349 671/350 132/350 514/350 512/351
227/351 522/351 527/352 466/352 530/352 196/353 432/353 139/354 138/354 225/354
650/355 665/356 130/357 197/357 534/358 538/359 202/360 136/360
lote 11: 601/365 644/365 648/365 608/366 653/366 214/366 646/367 221/367 560/367
572/367 667/368 571/368 668/368 570/368 228/369 573/369 583/369 581/369 582/370
212/370 427/371 525/371 532/372 217/372 140/372 422/373 229/374 679/374 224/375
586/376
lote 12: 389/391 622/391 623/392 388/392 624/393 468/393 332/393 392/393 331/394
330/394 469/394 394/395 390/395 391/395 625/395 626/396 393/396 470/397 627/397
628/397 472/397 401/398 473/398 629/398 402/399 554/399 557/399 556/400 555/400
397/400 398/401 396/402 254/402 471/403 268/403 257/404 249/405 251/405 250/406
252/407 266/407 258/408 260/408 269/409 400/410 253/410 399/411 395/411 267/412
255/413 265/413 318/413 319/414 317/414 591/414 590/415 272/416 271/417 270/417
288/418 632/419 295/419 294/420 290/420 637/421 477/422 474/422 478/423 293/423
631/424 630/425 403/425 404/426 298/427 480/427 287/428 296/428 292/429 633/430
635/430 299/431 479/431 289/432 636/433 475/433 476/434 297/434 634/435 291/436
300/436 481/437 333/437 273/438 280/440 279/441 281/442 274/443 277/445 282/446
278/447 276/448 275/449 580/450 283/452 584/452 321/453 639/453 640/454 320/455
464/455 322/456 335/456 323/457 465/458 336/458 638/459 324/459 593/460 304/461
301/462 306/464 307/465 302/466 305/467 303/468 325/469 337/470 328/472 327/473
326/474 329/475 587/476 308/478 313/480 314/482 312/484 310/485 311/487 309/489
462/491 338/492 463/494 315/496 405/499 316/502 334/506
lote 13: 210/511 421/512 605/512 75/512 434/512 134/512 76/513 79/513 436/513
78/513 547/514 559/514 441/514 442/514 620/515 618/515 619/515 523/516 94/516
524/516 90/517 104/517 105/517 219/518 562/518 561/519 621/519 519/520
lote 14: 67/528 68/529 122/529 209/529 417/529 71/530 603/530 73/530 74/530
423/531 131/531 431/531 428/531 77/532 611/532 80/532 222/532 521/533 95/533
93/533 533/534 106/534 101/534 89/535 592/535 614/536 615/536 100/537 567/537
566/538 109/539 579/540

```
lote 15: 1/551 66/562 2/572 0/585 3/599
lote 16: 415/609 416/609 72/610 607/610 129/610 429/610 612/611 610/611 439/611
87/611 88/611 86/612 520/612 528/613 426/613 529/614 445/614 98/614 97/614 99/614
108/615 549/615 551/616 564/616 617/617 433/618 550/619 552/619
lote 17: 70/627 82/627 537/627 83/628 81/628 553/628 85/629 96/629 91/629 92/630
588/630 589/630 531/631 103/631 102/631 594/632 526/633 145/639 146/649
lote 18: 69/658 107/658 563/659 575/659 574/660
lote 19: 84/664 545/664 596/665 595/665 597/665 565/666 576/666 578/667 577/667
515/668
```