Paulo César Brochonski

UM SISTEMA PARA PROGRAMAÇÃO DA PRODUÇÃO DE MÁQUINAS DE COMPOSIÇÃO SMT

Dissertação apresentada ao Curso de Mestrado em Informática Aplicada da Pontifícia Universidade Católica do Paraná, como requisito parcial para obtenção do título de "Mestre em Ciências"

Orientador: Prof. Dr. Marco A. B. Cândido

Curitiba, 1999

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
Pró-Reitoria de Pós-Graduação, Pesquisa e Extensão

ATA DA SESSÃO PÚBLICA DE EXAME DE DISSERTAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA APLICADA DA PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ.

Exame de dissertação nº 006

Aos 16 dias do mês de julho de 1999, realizou-se a sessão pública de defesa de dissertação "**UM SISTEMA PARA PROGRAMAÇÃO DA PRODUÇÃO DE MÁQUINAS DE COMPOSIÇÃO SMT**", apresentada por Paulo César Brochonski, ano de ingresso 1996, para obtenção do título de Mestre em Ciências. A Banca Examinadora foi composta pelos seguintes professores:

MEMBROS DA BANCA	ASSINATURA
Presidente:. **Prof. Dr. Marco Antonio B. Cândido (PUC-PR)**	
Prof. Dr. Júlio Cesar Nievola (PUC-PR)	
Prof. Dr. Anselmo Chaves Neto (UFPR)	
Prof. Dr. Ricardo Miranda Barcia (UFSC)	

De acordo com as normas regimentais a Banca Examinadora deliberou sobre os conceitos a serem atribuídos e que foram os seguintes:

MEMBROS DA BANCA	CONCEITOS
Presidente:. **Prof. Dr. Marco Antonio B. Cândido (PUC-PR)**	A
Prof. Dr. Júlio Cesar Nievola (PUC-PR)	A
Prof. Dr. Anselmo Chaves Neto (UFPR)	A
Prof. Dr. Ricardo Miranda Barcia (UFSC)	A
Conceito Final	

Observações da Banca Examinadora

PPGIA PUCPR

Profº Júlio Cesar Nievola
Coordenador do Programa de Pós-Graduação em Informática Aplicada–PUC-PR

Rua Imaculada Conceição, 1155 – Prado Velho – CEP 80215-901 – Telefone: +55 41 330-1669 / + 55 41 330-1654
Fax: + 55 41 330- 1669 – Curitiba – Paraná – Brasil - E-mail:ppgia@rla01.pucpr.br

Agradecimentos

Gostaria de agradecer aos meus pais Anastácio (em memória) e Etelvina por todo o suporte e incentivo. E de coração a minha esposa Eni e aos meus dois filhos Pedro e Michael pela paciência e pelos finais de semanas que ficamos em casa.

E muito importante salientar o apoio que recebi da SIEMENS, tanto oportunidade de realizar o curso quanto pelo apoio financeiro recebido.

Agradeço em especial o Prof. Cândido, meu orientador, pela idéias e caminhos mostrados. Ao corpo docente do Mestrado em informática Aplicada não só pelo pronto apoio mas também pelo espírito de cooperação e coleguismo gerado, que muito estimula a pesquisa.

E finalmente a todos aqueles que não mencionei mas que de forma direta ou indireta contribuíram na realização deste trabalho

Muito obrigado

Resumo

As placas eletrônicas usadas em computadores, sistemas de comunicação eletrônica embarcada, computadores entre outros, estão utilizando a tecnologia de montagem de superfície SMT (*Surface Montage Tecnology*).

A produção cada vez mais orientada a atender necessidades específicas do cliente está causando a diminuição dos lotes de produção. Em pequenos lotes de produção o tempo de preparação das máquinas de composição SMT é muito grande, ocasionando grandes tempos de paradas de máquinas.

Reduzir os efeitos dos tempos de preparação em pequenos lotes de produção aumenta a produtividade das linhas SMT. Para atingir este objetivo foram tomados dois caminhos: a formação de grupos e o escalonamento da produção dos grupos.

A formação de grupos reúne placas similares em grupos e assim reduz os tempos de preparação pois entre placas de um mesmo grupos não haverá tempos de preparação das máquinas.

No escalonamento da produção os grupos são sequenciados de forma a minimizar o tempo total de produção. No sequenciamento da produção foram utilizadas abordagens ótimas (*Branch and Bound*) e aproximadas (*Simulated Annealing*), onde foi levada em consideração a possibilidade de se fazer a configuração *off-line* das mesas de alimentadores (dinamicamente dependente da seqüência das placas).

O objetivo deste trabalho é a otimização do uso das máquinas de montagem SMT. Para alcançar este objetivo foram aplicadas tecnologias de programação da produção de alto desempenho aplicadas a um problema real da industria eletrônica.

ABSTRACT

Electronic boards are in use in many products as: communications systems, shipped electronic, computers... . This boards are using the assembly technology SMT (Surface Montage Technology).

SMT assembly lines are gradually replacing conventional printed circuit boards production lines. Manufacturing oriented to the client's specific need has led to smaller lot sizes and higher set up costs.

Minimise the set up effects in small batch sizes increases the productivity of SMT lines. The solution of this problem was achieved into two steps: group formation (Group Technology) and group scheduling.

Group formation involves the calculation of similarity coefficients between boards and a clustering procedure. Among boards of the same group there are no set up times.

The group scheduling determine the sequence of groups that minimises the makespan. To scheduling the groups were used optimal approaches (branch and bound) and non-optimal approaches (simulated annealing). The possibility of set up off line were considerate. The set up sequence is dynamically depend of the group sequence.

The goal of this work is optimize the use of SMT machinery. To reach the goal the were used high performance tools applied in a real problem of electronic industry

Índice

Índice de Figuras

Índice de Tabelas

1 Sistema para Programação da Produção de Máquinas SMT

1.1 Introdução

A tecnologia SMT (*Surface Montage Tecnology*) está presente desde o final da década de 80 no mercado. Esta nova tecnologia de montagem de placas eletrônicas permitiu a miniaturização dos produtos eletrônicos e ao mesmo tempo impôs mudanças estruturais nas linhas de montagem. A tecnologia SMT está substituindo os processos de montagem convencionais. Estima-se que até o ano 2000 o percentual de componentes com tecnologia SMT em uma placa eletrônica seja superior a 80%.

Os componentes utilizados na montagem SMT são chamados de componentes SMD (*Surface Montage Device*) , (Figura 1).

Os investimentos necessários para implantação de uma linha de montagem SMT (Figura 2) são de grande monta e o ciclo de evolução destas máquinas é

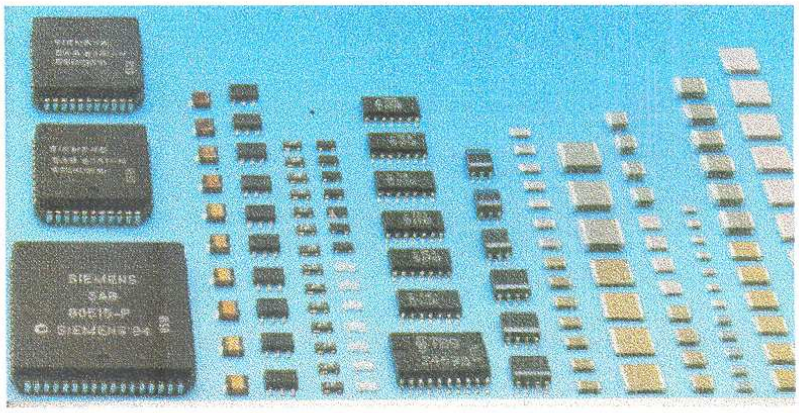


Figura 1. Componentes SMD

muito rápido, uma vez que componentes com novos encapsulamentos são introduzidos diariamente no mercado. Por estas razões, atualizações das linhas de montagem são necessárias a cada três anos aproximadamente.

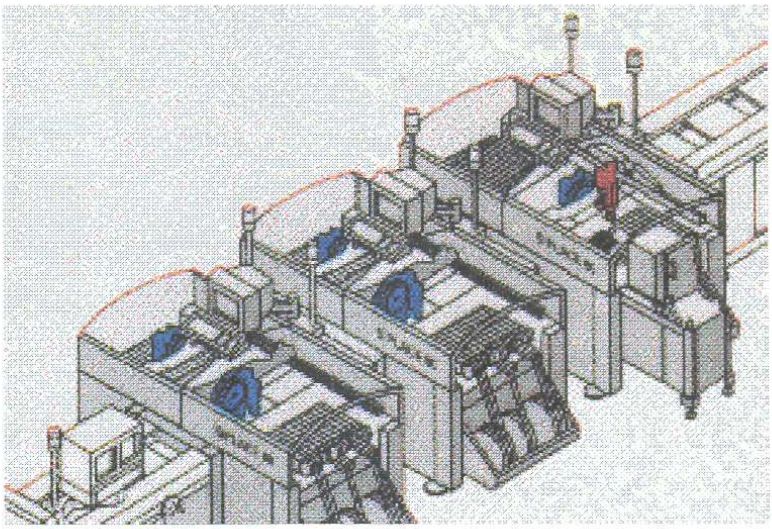


Figura 2. Máquinas de composição SMT

Portanto, por questões de amortizações de investimentos, é necessário extrair o máximo de produtividade das linhas de produção implantadas.

Outra característica do mercado é a constante evolução dos produtos. Empresas competitivas têm mais de 60% do seu faturamento advindos de produtos com menos de dois anos de vida. Com a necessidade das empresas conquistarem novos mercados, elas foram obrigadas a dar cada vez mais atenção a voz do mercado. A premissa atribuída a Henry Ford do cliente poder ter um carro de qualquer cor desde que ela seja preta está em desuso. Produtos cada vez mais orientados aos clientes, atendendo necessidades regionais, estão mudando as características das linhas de produção que eram baseadas na fabricação de grandes lotes.

O paradigma dos grandes lotes de produção está sendo modificado. A tendência é pela produção de lotes cada vez menores e mais diversificados. A conseqüência deste processo nas linhas de montagem SMT é uma grande diversidade no tipo de placas eletrônicas montadas por máquinas SMT. Assim tem-se um número grande de tipos de placas eletrônicas com lotes de produção pequenos.

Estes fatores implicam em linhas de produção ágeis e flexíveis a novos produtos, sem grandes tempos de preparação. Em outras palavras, um menor tempo de preparação das máquinas é um fator de incremento de competitividade das empresas.

Neste capítulo serão abordados os principais componentes de uma linha de montagem SMT, suas funções, a inserção do problema de programação com capacidade finita (*scheduling*) dentro do contexto de planejamento e controle da produção, a influência dos tempos de preparação na montagem de placas eletrônicas, e os ganhos que a otimização destes tempos podem trazer.

1.2 Montagem SMT

A montagem de placas eletrônicas utilizando a tecnologia THT (montagem através de furos) está sendo substituída pela técnica SMT. Esta técnica introduziu novos processos na indústria eletrônica. Uma visão geral dos principais componentes de uma linha de montagem SMT está apresentada na Figura 3.

A linha de montagem SMT é composta basicamente de três processos:

1 – serigrafia;

2 - composição dos componentes;

3 – solda.

A seguir descreveremos com mais detalhes cada uma das operações necessárias para a montagem de uma placa de circuito impresso (PCI) em uma linha SMT.

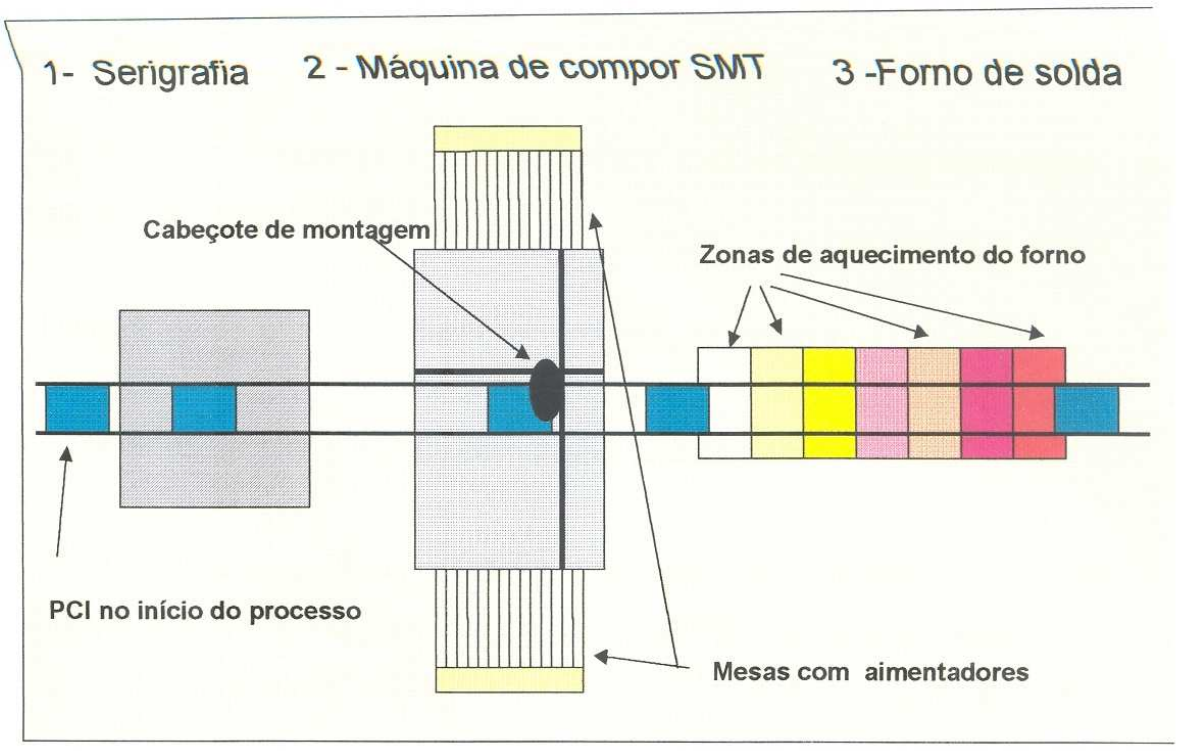


Figura 3. Diagrama linha de Montagem SMT

1.2.1 Serigrafia

A serigrafia está no inicio da linha de produção SMT. Por um processo serigráfico é passada pasta de solda nas ilhas de solda das placas de circuito impresso (PCI). Esta pasta é formada por estanho e um material que dá à pasta as características de um adesivo que se evapora com o tempo. Com a evaporação a pasta torna-se ressecada. O tempo de evaporação deste adesivo depende do tipo de pasta de solda. Este tempo é da ordem de uma hora. Na montagem do componente a pasta tem duas funções:

1. manter o componente preso a placa quando da montagem. Se a pasta estiver ressecada o componente pode sair da posição em que foi composto;

2. quando derretido pelo forno de solda, esta pasta faz o contato elétrico e a fixação mecânica entre o componente e a PCI;

1.2.2 Composição dos componentes

Com relação a velocidade existem diversos tipos de máquina de compor, entre os quais podemos distinguir:

- máquinas de alta velocidade com *menor* precisão;

- máquinas de alta precisão com menor velocidade;

- máquinas mistas. Alguns fabricantes oferecem máquinas mistas com dois cabeçotes de montagem, um de alta precisão para componentes fine-pitch[1], e outro cabeçote composto por um "revolver" que apanha vários componentes simples (resistores, capacitores, diodos,...) de uma só vez para aumentar a velocidade de composição.

A definição dos tipos de máquinas de uma linha é dependente do produto que ela vai montar. Quando há mais de uma máquina em linha pode ocorrer da primeira máquina terminar o ciclo de operação antes da máquina seguinte, ocasionando tempos de espera. A máquina que determinará a velocidade da linha será a mais lenta. Para solucionar este problema deve haver um balanceamento da carga de trabalho das máquinas de forma a maximizar a produtividade da linha (ARNOW, 1993).
O presente trabalho aborda linhas com uma única máquina. Estas linhas, embora mais simples, estão presentes em diversos sistemas reais de montagem SMT. Entretanto, pesquisas futuras devem ser feitas a fim de solucionar o problema de linhas com múltiplas máquinas.

Os componentes SMD são acondicionados em fitas, que podem ser de papel ou plástico. Estas fitas são enroladas em carretéis. Os carreteis são montados em dispositivos mecânicos chamados alimentadores. Os alimentadores com os carretéis de componentes são então posicionados nas mesas alimentadoras.

---

[1] Componente de alta precisão com pequeno espaçamento entre os terminais de contato

Os alimentadores têm como função deixar o componente SMD preparado para a composição. Assim que o cabeçote apanha um componente para montagem, o carretel avança a fita de componentes até que o próximo componente esteja na posição de composição.

A alimentação de placa nas máquina de composição SMT pode ser:

- **automática**: as placas entram na máquina de compor a através de esteiras oriundas da máquina de serigrafia (Figura 3);

- **manual**: um operador retira a PCI da serigrafia e alimenta as máquinas de SMT;

Neste trabalho tomou-se como base uma linha de média velocidade, capaz de montar 7000 componentes por hora (cph). Este tipo de máquina é normalmente usado para montagem de pequenos lotes de PCI. Os resultados obtidos podem ser estendidos a qualquer linha de montagem, bastando apenas construir interfaces de entrada e saída de dados específicos para a máquina em questão.

Na máquina de composição SMT as placas são transportadas até o local de montagem através de esteiras. Câmeras de vídeo no cabeçote localizam as marcas fiduciais na PCI. Através destas marcas é feito um reconhecimento da posição da placa nos eixos X e Y. Estes valores são utilizados para corrigir matematicamente pequenos desvios de posicionamento.

O programa de montagem da placa informa à máquina quais componentes montar; as coordenadas X, Y, Z onde o componente deve ser posicionado na placa; e em quais mesas (direita ou esquerda) o alimentador com o componente a ser montado está localizado. Com estas informações, o cabeçote se desloca até a mesa de alimentadores de componentes, apanha o componente por intermédio de uma pipeta que suga o componente, transporta

o componente até a placa e compõe o componente. Este ciclo se repete até que todos os componentes da placa estejam montados.

Existem máquinas onde o cabeçote de composição é fixo. Neste caso, a placa que está sendo montada e a mesa de alimentadores são móveis CRAMA at el. (1990). No presente trabalho são utilizadas máquinas em que o cabeçote de montagem é móvel e a placa e os alimentadores são fixos durante a montagem.

1.2.3 Solda

Ao sair da máquina de montagem, a PCI segue até o forno de refusão para ser feita a soldagem. O forno de refusão é dividido em várias zonas de temperatura. A temperatura de cada zona pode ser ajustada de modo a obter-se uma curva específica para cada PCI. A placa entra no forno com a temperatura ambiente e gradativamente vai sendo aquecida até o ponto de fusão do estanho. A placa é então resfriada, saindo do forno com temperatura próxima da ambiente.

A curva de temperatura aplicada à placa no trajeto dentro do forno determinará a qualidade da solda. Essa curva é dependente das características da PCI.

1.3 Tempos de preparação

Quando está se montando um produto com uma linha e vai se iniciar a montagem de um outro produto, existe a necessidade da preparação da linha de montagem para fabricação do novo produto.

A preparação da linhas SMT envolve alguns processos, tais como:

- Troca de alimentadores nas mesas;

- Troca de programas;

- Ajustes de máquinas;

- Troca das mesas de alimentadores;

- Inspeção da primeira peça produzida.

Estes processos demandam tempo e a minimização destes tempos de preparação das linhas de produção é um problema presente desde os primórdios da produção seriada. Várias soluções foram adotas, como por exemplo: linhas fixas de produção para produtos de grande demanda, lotes mínimos de produção ou aumento do número de mesas de alimentaoores. Mas estas soluções limitam a flexibilidade e podem aumentar os custos relativos a estoques.

Na indústria eletrônica, as máquinas SMT podem montar várias placas eletrônicas praticamente sem tempo de preparação entre as placas, desde que todos os componentes necessários para todas as placas estejam presentes nas mesas alimentadoras.

Em fábricas que têm um grande volume de produção sem uma variação grande no mix de produtos, o tempo de preparação é praticamente desprezível. Mas a realidade do mercado mundial está exigindo produtos mais exclusivos e dedicados. Como conseqüência, temos um aumento na diversidade das PCI's a serem montadas e uma redução nos tamanhos dos lotes de produção. Por estes motivos, a minimização do tempo de preparação das máquinas tornou-se um fator de ganho de produtividade, tão importante que em algumas fábricas foram colocados visores para monitorar o tempo gasto na preparação das linhas de montagem SMT.

Para montagem de placas de circuito impresso com a técnica SMT é necessário preparar as mesas alimentadoras com os componentes necessários na placa a ser montada. Para melhor visualizar o potencial de otimização que

pode ser obtido com a redução dos tempos de preparação, elaborou-se o exemplo abaixo:

Ex.: Calcular o *makespan* para montar um lote de 50 PCI's, sendo dados:

- cada PCI tem 100 componentes distintos;
- o tempo de preparação de cada componente na mesa é de cinco minutos;
- o tempo de troca de programas de produção é de 20 minutos;
- o tempo para montagem de cada placa é 1 minuto;

Tabela 1. Tempos de preparação e execução.

	Tarefa	Parcial	Total (minutos)
1	Tempo de montagem de cada placa é de 1 minuto	50 placas* 1	50
2	Cinco minutos/alimentador (retirar + colocar o componente):	100 comp. * 5	500
3	Tempo de troca de *setup* de 20 minutos (carga de programa + troca de mesas)	20	20
		Total	570

Assim, tem-se:

Com pode ser observado na tabela 1 o tempo total de *setup* é de 520 minutos e o tempo de montagem é de 50 minutos. Ou seja, a montagem leva um tempo inferior a 10% do tempo de *setup*. Neste caso, as máquinas de composição SMT ficariam paradas 90% do tempo aguardando *setup*.

1.4 Otimização do tempo de preparação

Existem várias formas de minimizar o tempo ocioso das máquinas SMT. Entre elas, se destacam:

1. utilização de linhas de montagem dedicadas para cada produto (ou para cada família de produtos que utilizam o mesmo *setup*);

2. utilização de máquinas que permitem a preparação *off-line*, isto é, enquanto a máquina está produzindo um lote, outro está sendo preparado;

3. aumento no número de alimentadores e mesas disponíveis;

4. aumento no número de pessoas destinadas a preparação das máquinas;

5. racionalização do número e do tempo de preparações através de agrupamento de placas similares e do seqüenciamento destes grupos pelas linhas;

Este trabalho aborda a quinta alternativa, em um ambiente que utiliza máquinas que permitem preparação *off-line* (2ª alternativa). Usando esta combinação pretende-se tirar proveito das similaridades existentes entre as placas para reduzir o tempo de *setup*.

1.5 Visão Geral do Planejamento, Programação e Controle da Produção

Equipamentos que necessitam de placas eletrônicas usualmente não são simples unidades. Eles são compostos de vários componentes e sub-montagens intermediárias, alguns fabricados e outros comprados. Na estrutura hierárquica do planejamento da produção são especificados os recursos e ações necessárias para atender à demanda dos produtos.

O PCP (Planejamento e Controle da Produção) visa gerenciar o sistema de produção através da integração eficaz dos fluxos de materiais usando sistemas de informações (CANDIDO,1998). O PCP é modernamente inserido no

gerenciamento integrado da cadeia de suprimentos. A cadeia de suprimentos usualmente é formada pelos seguintes componentes: fornecedores, estoque de matéria prima, fabricação e montagem, estoque de produtos acabados, distribuidores, consumidores, pós-venda (Figura 4).

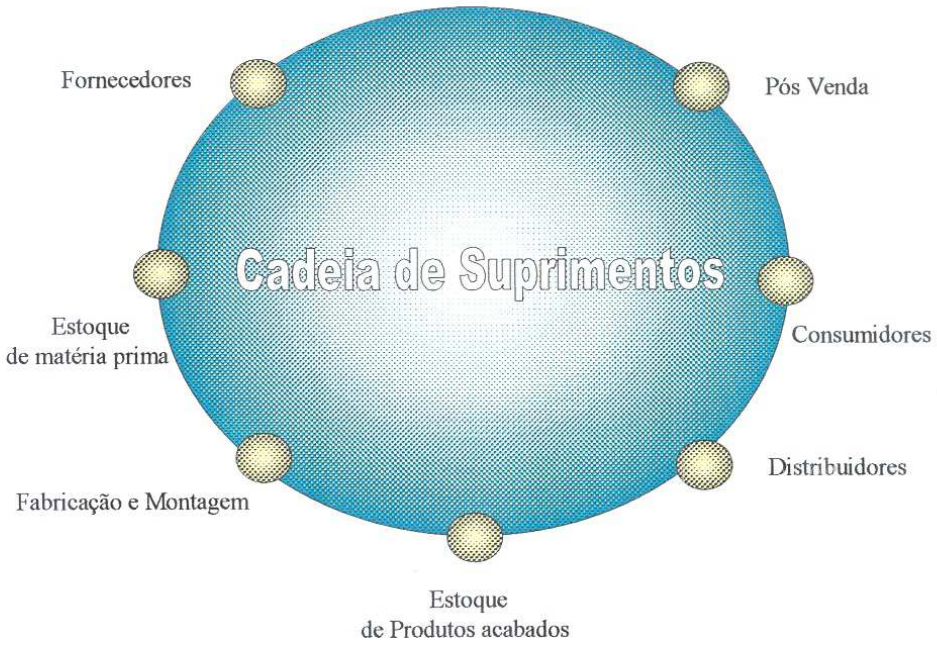


Figura 4. Cadeia de suprimento

Os componentes do planejamento da produção podem ser visualizados na Figura 5. O planejamento de produção é um processo dinâmico e deve estar sempre sendo realimentado com informações oriundas das etapas do processo, e assim aumentar a confiabilidade das saídas do sistema (CANDIDO,1998).

A cadeia logística do PCP envolve basicamente o planejamento agregado de longo e médio prazo, o MRP (*Material Requirement Planning*), o planejamento de capacidade (global e detalhado), a programação e o controle do chão de fábrica. O sistema PCP é composto dos seguintes elementos:

- Previsão de demanda;
- Planejamento agregado de longo e médio prazo;

- Plano Mestre de Produção (MPS);
- Planejamento das Necessidades de Materiais (MRP);
- Planejamento de Capacidade:
  - planejamento global (grosseiro);
  - planejamento detalhado;
- Programação da Produção com Capacidade Finita (*scheduling*);
- Controle da Produção.

O planejamento e controle da produção é um processo hierárquico tradicionalmente dividido em curto, médio e longo prazo. Cada horizonte de planejamento tem objetivos e restrições específicas (CANDIDO,1998). Divisões do planejamento hierárquico:

1) longo prazo (estratégico): 3 a 10 anos com atualização anual;

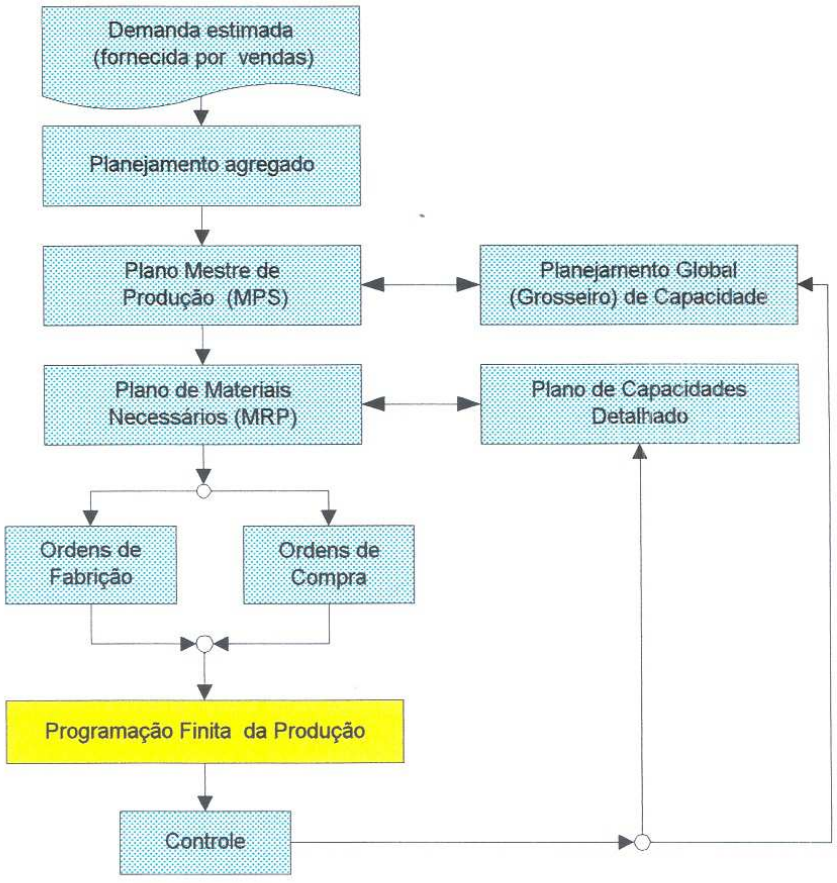


Figura 5. Planejamento da Produção

2) médio prazo (tático): 6 meses a 2 anos com atualizações mensais;

3) curto prazo (operacional): 1 semana a 6 meses. Esta dissertação está inserida no planejamento operacional.

O sistema de planejamento da produção, como um todo, tem as seguintes **entradas básicas:**

- Previsões de demanda (em ambientes com fabricação para estoque) e/ou carteira de pedidos (em ambientes com fabricação por pedidos);
- Listas de materiais para cada produto;
- Planos de processamento;
- Situação do estoque;
- Capacidade produtiva (maquinário e mão-de-obra);

Saídas básicas do planejamento da produção:

- Plano de produção viável;
- Plano de compras viável.

A seguir estão descritos os principais elementos de um sistema de PCP

1.5.1 Planejamento Agregado

O Planejamento agregado permite uma análise global da estratégia da empresa. Trata do planejamento de médio e longo prazo de vendas, produção, compras, estoques, custos e capacidades.

Os passos básicos para elaboração do planejamento agregado são:

- Obtenção de um plano de vendas a médio de longo prazo, agregado em nível de grupo ou família de produtos. Este plano agregado pode ser obtido, por exemplo, através de modelos de previsão;

- Obtenção de um plano de produção pela utilização de planilhas ou modelos de programação matemática;

- Desagregação: obtenção de um MPS - Plano Mestre de Produção desagregado em nível de produto acabado (ou semi-acabado em certos casos);

1.5.2 Plano Mestre de Produção (MPS)

O plano mestre da produção estabelece quais os produtos finais que serão feitos, em que datas e em que quantidades. O MPS depende da natureza do ambiente produtivo que pode ser:

- fabricação por pedidos (*make-to-order*);
- fabricação para estoque (*make-to-stock*);
- montagem por pedidos (*assembly-to-order*);

Além da demanda estimada pela previsão o MPS também incorpora a demanda de outras fontes tais como carteira de pedidos de clientes, e necessidades de estoques de segurança. O horizonte de planejamento do MPS é usualmente de algumas semanas até seis meses MOREIRA (1998).

1.5.3 Planejamento de Capacidade

Planejamento de capacidade ocorre em dois momentos:

1. no planejamento agregado e no MPS. O planejamento de capacidade é chamado de Planejamento Global (ou Grosseiro) de Capacidade (*Rough Cut Capacity Planning* - RCCP). O RCCP é uma comparação entre a carga horária produtiva exigida pelo Plano Agregado, ou pelo MPS, e a carga disponível nos departamentos (ou centros de trabalho). O RCCP é uma

avaliação inicial e grosseira da capacidade produtiva que auxilia na determinação da viabilidade do Plano Agregado (CANDIDO,1998);

2. no MRP é chamado Planejamento Detalhado de Capacidade (*Capacity Requirement Planning - CRP*). Onde é feita uma verificação mais detalhada da capacidade produtiva.

1.5.4 Planejamento das Necessidades de Materiais – MRP

O MPS é explodido para as sub-montagens e componentes da BOM (Bill of Material) do produto acabado, gerando uma série de sugestões de compra e produção pelo MRP (*Material Requirements Planning*), que teve origem nos meados dos anos 60. A função principal do MRP é garantir a disponibilidade de materiais. O MRP determina as quantidades e datas em que cada material deve ser produzido ou adquirido externamente, a fim de atender às demandas do mercado.

Este processo envolve a monitoração de estoques e a criação automática de sugestões de ordens de produção e compra.

Entradas do MRP:

- Necessidades "independentes" e pedidos de clientes dos produtos acabados;
- Situação do estoque dos materiais;
- Ordens de produção e de compra já confirmadas (recebimentos confirmados);
- "Listas de materiais" (BOM) contendo a estrutura de cada material;
- *Lead times*: tempos de produção para materiais fabricados internamente e prazos de fornecimento para materiais comprados de terceiros;
- Estratégias de planejamento (*make-to-stock, assemly-to-order*, etc.);
- Outros: política de determinação de tamanho de lote, níveis de estoques de segurança, índice de refugo, etc (MOREIRA,1998).

Saídas do MRP:

- conjunto de sugestões de ordens de produção no horizonte de planejamento;
- conjunto de sugestões de ordens de compra no horizonte de planejamento;

O funcionamento do MRP é baseado em quatro passos:

1. Explosão: simulação da explosão do produto em subcomponentes e matérias-primas. Para realizar esta explosão é necessário o conhecimento da estrutura do produto, que é a sua lista técnica,

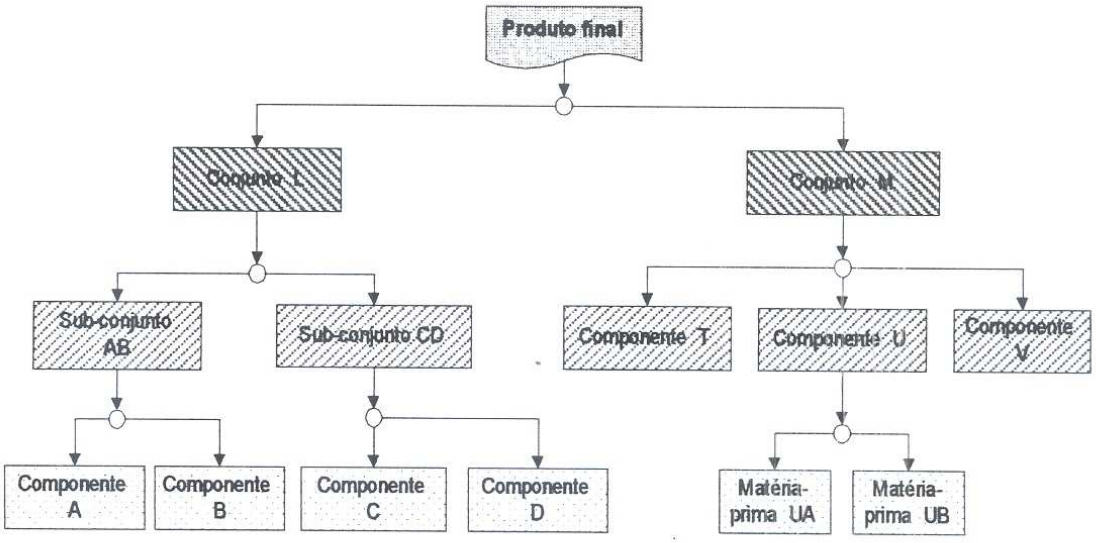


Figura 6. BOM (*Bill of material*)

conhecida como BOM (*Bill of Material*), como mostra a Figura 6.

2. Determinação das necessidades líquidas: ajuste das necessidades brutas para acomodar o estoque disponível e as ordens de produção e compra já programadas;

3. Determinação dos tamanhos dos lotes: determinação dos tamanhos dos lotes das ordens sugeridas de produção e compra (que são geradas para cobrir as necessidades líquidas);

4. Programação de prazos: determinação das datas de início e fim das ordens sugeridas, por meio dos *lead times* de fabricação e dos prazos de entrega dos fornecedores (MOREIRA,1998).

1.5.5 Planejamento dos Recursos de Manufatura - MRPII

O MRP II é uma evolução do MRP e incorpora, entre outras funcionalidades, funções de Planejamento de Capacidade.

Diferente dos sistemas JIT/Kanban que são sistemas de "puxar", o sistema MRP II são geralmente sistemas de "empurrar". Atualmente estes sistemas encontram-se embutidos em sistemas de ERP (Enterprise Resource Planning).

Principais pontos fracos do MRPII:

- Assume capacidade de produção infinita;

- Exige dados acurados de todos os eventos do chão de fábrica para o cálculo dos tempos de percurso (*lead times)*;

- Assume *lead times* fixos (médias históricas), o que pode gerar programações inviáveis ou nível desnecessariamente baixo de utilização dos recursos;

- Não é apropriado para programação e controle. Daí a necessidade de um *scheduler* complementar, onde parte desta dissertação se insere (MOREIRA,1998).

1.5.6 Programação da Produção com Capacidade Finita (Scheduling)

Programação da produção (scheduling) é o problema de determinar uma alocação do recursos produtivos no tempo para executar um conjunto de ordens de produção, a fim de minimizar uma medida de custo.

Sistemas de *scheduling* visam minimizar os níveis de estoque, maximizar a utilização dos recursos e os índices de produtividade, e melhorar o cumprimento de datas de entrega.

Entradas básicas:

- Ordens de Produção (sugeridas ou já confirmadas);
- Planos de processamento de cada produto;
- Capacidade de cada centro de trabalho.

Saídas básicas: Alocação viável dos recursos produtivos no tempo para executar as operações das ordens de produção, minimizando uma medida de custo;

Os problemas de scheduling podem ser classificados de acordos com diversos critérios:

- Métrica de avaliação;
- Dinâmicos x Estáticos;
- Estocásticos x determinísticos;
- Segundo fluxo / complexidade do ambiente.

O *scheduling* está intimamente ligado com o nível operacional da produção (chão de fábrica). Na programação da produção com capacidade finita são planejados a ocupação das máquinas, operadores, ferramentas. O foco desta dissertação esta nesta fase (Candido, 1998).

1.5.7 Controle da Produção

O controle da produção diz respeito ao acompanhamento das ordens de produção e à realimentação do sistema de PCP com eventos que ocorrem no chão de fábrica. Especificamente, a função do Controle da Produção é controlar o processamento de uma ordem de produção, que engloba:

- criação da ordem de produção;
- verificação de disponibilidade de material e capacidade para a ordem;
- liberação e impressão da ordem;
- retirada de material do estoque para ordem;
- confirmação da ordem (durante/após a execução);
- recebimento no estoque do material produzido pela ordem;
- cálculo de desvios (custo real vs. estimado) e liquidação da ordem no final do período.

1.6 Motivação do Trabalho

Este trabalho foi motivado pela necessidade do aumento da produtividade das máquinas de montagem SMT. Desta maneira, espera-se incrementar as vantagens competitivas da empresa, pois no mercado globalizado só sobreviverão as empresas que mantiverem uma vantagem competitiva sobre seus concorrentes MARTINS (1998).

As melhorias advindas da introdução de um sistema de programação da produção com capacidade finita que leve em conta a similaridade entre as placas para redução do *setup* podem ser enquadradas nos seguintes critérios:

- **Custos** : ganhos de custo podem ser atingidos pela melhor utilização do ativos (máquinas e alimentadores), evitando assim a aquisição de novos equipamentos, reduzindo o repasse de custos de aquisição e amortização sobre o produto final. Além disso, pretende-se melhorar ocupação da mão de obra na preparação das máquinas, evitando trabalhos desnecessários;

- **Qualidade** : Com uma seqüência de produção otimizada, reduz-se o número de preparações necessários. Com um número menor de *setups*, o processo fica mais estável e menos vulnerável a erros;

- **Prazos de entrega**: as ordens de produção de produção são normalmente agendadas por ferramentas como o MRP. Para determinar o prazo de entrega do produto estas ferramentas utilizam *lead times* médios, que não permitem a determinação exata dos prazos de entrega. Com o novo sistema, espera-se uma maior precisão nas datas de entrega;

- **Flexiblidade**: caso seja necessário fazer alterações no programa de produção, o gerente pode utilizar o sistema proposto para fazer simulações, analisar as conseqüências das mudanças e tomar a decisão com base em dados.

1.7 Objetivo do trabalho

Este trabalho se propõe a implementar um sistema para melhorar a produtividade de uma linha de montagem SMT.

Trata-se de um sistema para resolver um problema programação da produção com capacidade finita de uma linha de produção SMT com uma máquina, quatro mesas alimentadoras, *n* placas e *m* componentes por placa onde se conhece:

- a demanda por cada placa;

- a configuração de cada placa;

- a capacidade de alimentadores por par de mesas, *fc.* Vale lembrar que há duas mesas conectadas na máquina e duas em processo de *setup off-line.*

Há um operador de máquina, e um preparador de mesa na área de preparação;

- os tempos de montagem e desmontagem dos alimentadores nas mesas;

- os tempos médios de montagem de cada placa. Foi desconsiderado o efeito da distância do alimentador em relação a posição do cabeçote de montagem, uma vez que este fator tem uma influência mínima na produção em pequenos lotes;

- o fato de todos os alimentadores utilizados serem de oito milímetros simples;

- o tipo de máquina utilizada, que movimenta o cabeçote e mantém fixos a placa e os alimentadores durante a montagem;

Portanto, o objetivo é determinar uma seqüência de produção de placas (e as configurações das mesas alimentadoras correspondentes) de modo a minimizar o tempo total de produção. Trata-se de um problema NP-completo.

Como será descrito nos capítulos seguintes foram utilizadas técnicas de formação de grupo, e técnicas ótimas e aproximadas para programação finita da produção.

1.8 Organização do Trabalho

Esta dissertação está organizada da seguinte forma: no capítulo dois é apresentada uma revisão da literatura que abrange a tecnologia de grupos na área de SMT e as técnicas de programação da produção com capacidade finita. No capítulo três descrevemos a entrada de dados e a estrutura geral do sistema proposto. A formação de grupos está detalhada no quarto capítulo. Os algoritmos de *branch and bound* e *simulated annealing,* para a programação da

produção com capacidade finita, estão no quinto capítulo. Finalmente no sexto capítulo são descritos os resultados alcançados e traçadas as conclusões e sugestões para futuras pesquisas na área.

2 Principais abordagens utilizadas para tratar problemas de formação de grupo e programação da produção com capacidade finita (scheduling)

2.1 Introdução

Um grande número de trabalhos sobre otimização em linhas de montagem SMT foram apresentados nos últimos anos, utilizando várias abordagens: balanceamento de linha (ARNOW, 1993), programação linear, métodos ótimos, regras de despacho, tecnologia de grupos (SHTUB at al., 1991; SHTUB at al.,1992; e FÖHRENBACH, 1997), lógica fuzzi (GÜNTHER at al., 1997), técnicas de busca local, (CRAMA at al.,1990; e GRONALT at al.,1997) dentre outros. Esta literatura foi valiosa para a formação da estrutura deste trabalho. Entretanto, vários desses artigos e livros são específicos para determinados tipos de máquinas (CRAMA at al.,1998); outros são muito teóricos e não contemplam aplicações práticas; e muitos ficaram desatualizados em função da rápida evolução tecnológica.

A otimização da produção foi dividida em dois sub-itens:

1. formação de grupos;

2. programação de uma linha de produção com capacidade finita da produção (*scheduling*).

Neste capítulo, maneiras distintas de resolver estes dois itens serão analisadas e abordadas de forma resumida. Pelo fato da programação da produção para o caso tratado aqui ter uma analogia com o clássico problema do caixeiro viajante, este tema também será abordado.

2.2 Otimização em montagem SMT (revisão bibliográfica)

A maximização da produtividade em linhas de montagem SMT tem se mostrado um campo fértil de pesquisa na década de 90, um vez que a tecnologia SMT é nova e tem grande potencial de crescimento.

O planejamento e a montagem de várias placas distintas em um linha de SMT, utilizando máquinas do fabricante FUJI e envolvendo também o balanceamento de linha são discutidos em CRAMA at al. (1998) e CRAMA at el. (1990). A utilização de *setups* parciais na troca de placas de circuito impresso é abordada por LEON & BRETT (1994). Em SHTUB at al. (1991) e SHTUB at al. (1991) a tecnologia de formação grupos é utilizada na montagem de placas de circuito impresso. A lógica fuzzi é usada na otimização de *setups* no trabalho de GÜNTHER at al. (1997). A utilização de lógica difusa na agregação de placas com pequenos lotes de produção também é explorada em FÖHRENBACH (1997). O seqüenciamento de componentes em fitas para montagem SMT está formulado em SULE (1996). A troca de componentes em máquinas SMT é abordada por GRONALT at al. (1997). O aumento da produtividade de linhas de produção SMT através do balanceamento das linhas foi tratado em ARNOW (1993). Um problema com seqüências dependentes de *setup* é abordado por CANDIDO at al. (1998b).

2.3 Tecnologia de grupos

A tecnologia de grupos (GT) está baseada no principio de que "coisas similares devem ser feitas de maneira similar". Assim, o agrupamento de componentes ou produtos que possuem características similares, ou ainda que requeiram processos de fabricação de mesma natureza, pode vir a ocasionar ganhos substanciais na produtividade.

As similaridades entre os objetos em questão podem ser exploradas em muitas áreas, incluindo o desenvolvimento de produtos, o projeto de *layout*, etc. As similaridades entre as peças também pode ser utilizada para formar famílias de

peças similares e, desta forma, facilitar a programação da produção. A GT é um dos princípios básicos da manufatura celular.

Três etapas básicas estão envolvidas na utilização da tecnologia de grupo:

Codificação: é a descrição alfanumérica das peças. Este código denominará a peça em questão. O código é uma combinação alfanumérica e poderá conter uma ou mais das seguintes informações: desenho, forma, tamanho, tipo de material, características técnicas, processos envolvidos na fabricação. Neste trabalho o código utilizado é a identificação dos componentes que serão assentados na placa de circuito impresso.

Classificação: também conhecida por formação dos grupos, refere-se ao uso dos códigos e outras informações sobre as peças para o agrupamento de peças que requerem processos similares em famílias (ASKIN at al., 1993). Uma família de peças deve ser processada em uma "célula de manufatura" . A célula deve conter todos os equipamentos necessários para a fabricação de uma ou mais famílias de peças.

Layout: está relacionado com a disposição física dos recursos de manufatura. Neste trabalho existe apenas uma linha, composta de uma máquina. Esta máquina permite a preparação off-line.

A utilização de GT simplifica a programação da produção com capacidade finita, pois as peças de um mesmo grupo (família) requerem os mesmos processos de fabricação, as mesmas máquinas, e usualmente não requerem *setups* adicionais. Assim, dentro do processo de produção, peças de um mesmo grupo devem ser, sempre que possível, produzidas na seqüência a fim de reduzir os tempos gastos na preparação das máquinas (MARTINICH, 1997).

Neste trabalho, a tecnologia de grupo é utilizada para agrupar PCIs com componentes similares. Se as placas que pertencem a um grupo forem montadas na seqüência não é necessário reconfigurar a máquina SMT, e desta forma é possível reduzir o tempo de *setup* da máquina. A amplitude desta

redução dependerá da similaridade entre as placas. Dois trabalhos específicos na formação de grupos de placas de circuito impresso foram utilizados como referência: SHTUB at al. (1991) e SHTUB at al. (1992).

Um fator que limita o número de placas de circuito impresso em cada grupo, é o número de alimentadores que a máquina comporta (CRAMA at al., 1998) . Se o número de alimentadores fosse ilimitado, haveria apenas um único grupo composto por todas as placas. Mas na prática é geralmente impossível incluir todas as placas em um único grupo.

Na a formação de grupos são utilizados coeficientes de similaridades. Estes coeficientes informam o quão similar as peças são. Vários coeficientes podem ser utilizados. Como exemplos tem-se o coeficiente apresentado por ASKIN at al. (1993), o MAXSC utilizado por Frörenbach (FÖHRENBACH, 1997), e o coeficiente de Jacard (SOKAL, 1963), utilizado em SHTUB at al. (1991) e também neste trabalho.

2.4 Programação da produção com capacidade finita (Scheduling)

A programação da produção com capacidade finita consiste em alocar os recursos no tempo para realizar um conjunto de tarefas, respeitando um conjunto de restrições, de forma a minimizar uma certa medida de custo.

Na manufatura industrial, o problema de *scheduling* envolve a utilização de um conjunto de recursos (tais como máquinas, ferramentas e mão de obra) para executar as operações de fabricação e montagem de uma série de produtos, com o objetivo de minimizar os custos de produção.

De uma maneira geral os problemas de *scheduling* são NP-completos, como descritos no item 2.4.1. Duas grandes classes de abordagens são utilizadas para a resolução deste tipo de problema: as abordagens ótimas e as abordagens aproximadas. As abordagens ótimas são utilizadas em problemas

de pequenos porte. Para problemas maiores, utilizam-se as abordagens aproximadas, que não chegam necessariamente a uma solução ótima, mas apresentam um bom resultado, com um tempo computacional razoável. Algumas destas metodologias serão abordadas adiante neste capítulo.

2.4.1 Problemas de Otimização combinatorial: Complexidade dos algoritmos

Resolver um problema de busca combinatorial é encontrar uma solução que satisfaz um conjunto de restrições entre um conjunto muito grande, mas finito (ou infinito mas contável), de possíveis soluções (KARP, 1986).

São problemas clássicos de combinação combinatorial: a programação de produção (*scheduling*), roteamento de veículos, o problema da mochila (*Knapsack*), layout de circuitos VLSI, caminho Euleriano, colocação de grafos, entre outros (CASTI, 1996).

Os problemas de busca combinatorial podem ser problemas de decisão ou otimização. A solução de um problema de decisão é simplesmente responder sim ou não a uma pergunta (ex.: existe um circuito Hamiltoniano para o grafo dado?). Um problema de otimização combinatorial requer encontrar uma combinação entre um conjunto das soluções que minimiza uma função custo (ex.: determine a programação da produção que minimiza o tempo total de produção). É sempre possível usar artifícios para transformar um problema de otimização em um problema de decisão. Estes artifícios permitem que todos os problemas de busca combinatorial sejam tratados como problemas de decisão sem que haja perda de características.

Para alguns problemas combinatoriais existem algoritmos disponíveis para resolvê-los em tempo computacional razoável (ex.: caminho Euleriano). Outros problemas são extremamente difíceis de resolver e podem requerer um enorme e impraticável tempo computacional (ex.: maioria dos problemas de *Scheduling).* Um problema é chamado de tratável se ele pode ser resolvido em

um número de passos limitado por um polinômio do tamanho de suas entradas e intratáveis caso contrário.

Informalmente, um problema está na classe P se ele for tratável. P é o conjunto de todos os problemas que podem ser resolvidos em um tempo polinomial.

Um problema de decisão D1 é redutível para um problema de decisão D2 se e somente só o problema D1 pode ser transformado em D2 por uma função de tempo polinomial. Portanto se D2 está em P então D1 também estará em P.

NP (*non–deterministic polinomial*) é a classe dos problemas combinatoriais que são verificados em tempo polinomial. Dada uma solução, é possível verificar se a solução é correta em um tempo polinomial. É claro que todos os problemas em P estão também em NP, isto é, $P \subset NP$. A questão se P=NP não foi ainda respondida. Contudo é muito provável que eles não sejam o mesmo conjunto. Se P = NP encontrar uma solução e verificar a solução apresentariam a mesma dificuldade e este conceito parece não ser verdade.

Um problema de decisão D é NP-completo se ele está em NP e se todos os problemas em NP são redutíveis a D. Isto significa que se um problema NP-completo for solucionado em tempo polinomial, então todos os problemas de NP também podem ser resolvidos em um tempo polinomial. Os problemas NP-completos são os problemas combinatoriais mais difíceis.

Heurísticas que encontram uma boa solução, mas não necessariamente a ótima, devem ser desenvolvidas para superar esta intratabilidade intrínseca dos problemas NP-completos. A maioria absoluta dos problemas de scheduling são NP-completos. A lista dos problemas NP-completos é grande, contendo centenas de problemas (GARAY & JOHNSON, 1979).

Uma formulação formal deste e de outros conceitos apresentados nesta seção está além do escopo desta dissertação. Eles podem ser encontrados em GARAY & JOHNSON (1979), bem como em CORMEN at al. (1990).

2.4.2 Caixeiro Viajante

O problema do caixeiro viajante provavelmente está entre os mais estudados na área de otimização discreta. Muitos problemas de programação de produção são análogos ao problema do caixeiro viajante (TSP - *Travelling Salesman Problem*).

O enunciado do TSP é o seguinte: um caixeiro deseja visitar várias cidades, passando por cada cidade uma única vez, e retornando para a cidade de origem. São conhecidos os custos associados a cada trajeto unindo duas cidades. O problema consiste em determinar a rota que visita todas as cidades e retorna à cidade de origem com o menor custo. Trata-se de um problema de busca NP-completo (CORMEN at al., 1990).

A solução ótima para um TSP de 3038 cidades foi conseguido através de computação paralela com cinqüenta estações de trabalho (BIXBY & APPLEGATE, 1993). Johnson and McGeoch reportam que uma rede de computadores levou entre três a quatro anos para solucionar um TSP de 7397 cidades (JOHNSON & MCGEOCH, 1997).

O TSP é simétrico quando os custos dos trajetos que conectam as cidades são iguais em ambos os sentidos, e assimétrico em caso contrário.

A taxa de crescimento do espaço de soluções do TSP é exponencial (igual ao fatorial do número de cidades). Para n=3, onde n é igual ao número de cidades a serem visitadas, teríamos seis caminhos distintos. Para dez cidades há mais de trez milhões de possibilidades, como mostra a Tabela 2. Encontrar uma solução através de uma busca exaustiva torna-se impraticável para problemas de grande porte. Com a programação dinâmica o problema pode ser resolvido em $O(n^2*2^n)$ (TERRADA, 1991). Várias soluções com as mais variadas técnicas já foram apresentadas para solucionar o TSP.

Um fórum de discussão do TSP é a TSPLIB, onde se encontram vários métodos e uma rica fonte de referências com vários problemas *benchmark* (REINELT, 1991).

Tabela 2. Taxa de crescimento TSP

	Busca exaustiva	*Programação dinâmica*
n	*n!*	$n^2 * 2^n$
1	1	2
2	2	16
3	6	72
4	24	256
5	120	800
6	720	2304
7	5040	6272
8	40320	16384
9	362880	41472
10	3628800	102400
11	39916800	247808
12	4,79E+08	589824
13	6,23E+09	1384448
14	8,72E+10	3211264
15	1,31E+12	7372800

2.4.2.1 Formulação Matemática do TSP

Muitas formulações matemáticas foram propostas para o TSP. Apresentamos a seguir uma formulação mais convencional para o problema (BRADLEY at al., 1977):

- seja *n* o número de cidades a serem visitadas;
- seja c_{ij} o custo do deslocamento da cidade i até j;

- faça x_{ij} =1 quando o caixeiro for da cidade i para j e 0 caso contrário.

$$\text{minimize} \quad \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ij} \qquad (1)$$

sujeito a :

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, 2, \ldots, n) \qquad (2)$$

$$\sum_{j=1}^{n} xij = 1 \qquad (i = 1, 2, \ldots, n) \qquad (3)$$

$$x_{ij} >= 0 \qquad (i = 1,2,\ldots, n ; j = 1,2,\ldots, n) \qquad (4)$$

A função objetivo (1) armazenará os custos para se viajar da cidade i até j; as restrições (2) a (4) garantem que o caixeiro visite cada cidade exatamente uma vez. Esta formulação permite a existência de sub-trajetos não conectados (Figura 7)

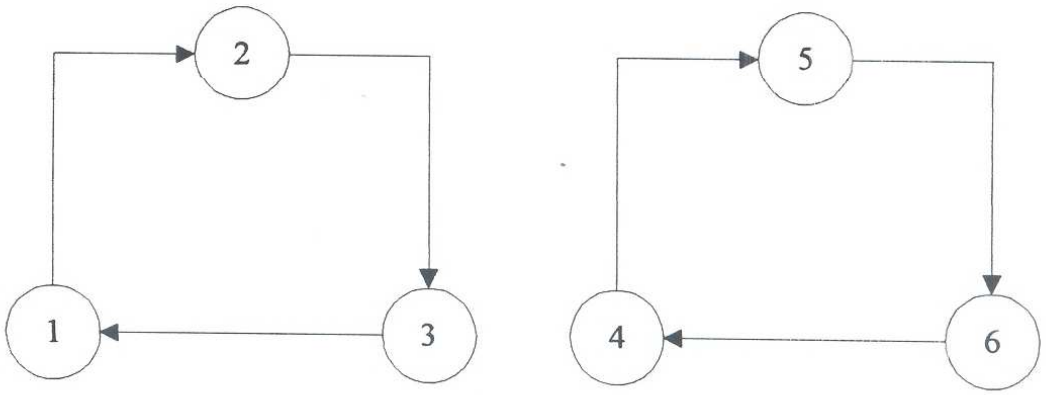


Figura 7. Sub-trajetos não conectados

Assim restrições adicionais devem ser incluídas para assegurar que não hajam sub-trajetos. Para evitar o sub-trajeto (Figura 7) , a seguinte restrição deve ser incluída:

$$X_{14} + X_{15} + X_{16} + X_{24} + X_{25} + X_{26} + X_{34} + X_{35} + X_{36} >= 1$$

Esta inequação assegura pelo menos uma rota entre as cidades (1, 2, 3) e as cidades (4, 5 ,6). Uma restrição semelhante a esta deve ser incluída para cada possibilidade na qual as cidades podem ser divididas em dois sub-trajetos não conectados. Para se eliminar os sub-trajetos haveria a necessidade de serem incluídas (2^n - 1) restrições. Assim o TSP torna-se um problema de programação inteira de grandes proporções (BRADLEY at al.,1977):

2.4.2.2 Analogia com o TSP

Se fosse possível montar uma tabela fixa dos *setups* necessários de um grupo de placas para outro, o *scheduling* dos grupos seria a resolução de um TSP tradicional. Mas o problema com o qual estamos trabalhando envolve *setups* dinâmicos, isto é, as distâncias entre as cidades se alteram cada vez que é realizado um movimento de uma cidade para outra. Além disto o problema contempla também o *setup off-line* que é um fator que torna o problema de *scheduling* em questão ainda mais complexo.

2.4.3 Abordagens ótimas para problemas de scheduling

As técnicas que apresentam soluções ótimas são utilizadas para resolver casos de problemas de programação da produção com capacidade finita (scheduling) com reduzido tamanho, devido a inerente intratabilidade da maioria dos problemas de *scheduling* (inclusive do estudado nessa dissertação). As abordagens ótimas mais utilizadas são os algoritmos de *branch and bound*, e a programação dinâmica.

2.4.3.1 Branch and Bound

A técnica de *Branch and Bound* (ramificar e limitar) é provavelmente a abordagem ótima mais amplamente usada em programação finita da produção.

O principio básico da técnica branch and bound é "dividir para conquistar". Uma vez que o problema original é muito difícil de ser resolvido diretamente, ele então é sucessivamente subdividido em problemas menores que podem ser conquistados. Esta divisão (*branching*) é feita particionando o conjunto de todas as soluções, em conjuntos de menores. A conquista (*fathoming*) ocorre quando:

1. o valor do limite inferior da função custo do ramo em questão for superior ao limite superior vigente;
2. todas as soluções que possam ser obtidas pela exploração do nó em questão corresponderem a soluções inviáveis, de acordo com testes específicos;
3. o nó corresponder a uma solução viável inteira;

Um fator que pode acelerar significativamente o tempo computacional do algoritmo é a existência de um limite superior inicial de alta qualidade. Este limite pode ser determinado através de uma heurística específica. Quanto mais próximo este valor estiver da solução ótima, menor será o tempo de busca.

2.4.3.2 Programação Dinâmica

Os fundamentos da programação dinâmica foram elaborados por Bellman na década de cinqüenta. Segundo BLAZEWICZ at al. (1994) o nome Programação Dinâmica não é muito adequado. Uma melhor descrição seria programação recursiva ou de múltiplos estágios. Isto significa que o problema é divido em estágios. A cada estágio uma decisão deve ser tomada. Esta decisão terá impacto nos estágios seguintes. A programação dinâmica pode ser utilizada quando a solução ótima puder ser obtida como o resultado de uma seqüência de decisões localmente ótimas (HOROWICZ at al., 1998).

Neste método, a recursividade de funções é utilizada para se chegar a uma solução ótima sem a necessidade de se enumerar todas as possibilidades (WAGNER, 1989). O sucesso desta técnica está relacionado à habilidade de

formulação do problema. A diferença entre uma formulação grosseira e uma bem elaborada determinará o tempo computacional para se chegar a uma solução.

Um exemplo da resolução do TSP através da programação dinâmica é apresentado em TERRADA (1991).

2.4.4 Abordagens aproximadas para problemas de scheduling

As técnicas aproximadas são utilizadas em problemas de programação da produção com capacidade finita quando, devido ao tamanho do problema, a solução ótima não pode ser encontrada em tempo razoável. As abordagens aproximadas ou heurísticas tem como objetivo encontrar uma boa solução, mas não necessariamente uma ótima.

Entre as técnicas mais utilizadas estão as regras de despacho, os algoritmos genéticos, a busca tabu, e o *simulated annealing* (Recozimento simulado).

2.4.4.1 Regras de despacho

As regras de despacho provavelmente estão entre os métodos mais populares para programação da produção. As regras de despacho determinam a próxima operação a ser processada dentre um conjunto de operações possíveis. As regras de despacho podem ser divididas em estáticas e dinâmicas.

As regras de despacho estáticas tem as prioridades definidas em um índice que se mantém constantes durante todo horizonte de programação. Ao contrário, as regras dinâmicas mudam com o tempo (ASKIN at al., 1993).

Entre as principais regras de despacho temos:

- SPT (Shortest Processing Time), seleciona a tarefa com o menor tempo de processamento;

- EDD (Earliest Due Date), seleciona a tarefa com a data de entrega mais próxima;
- FIFO (First-in, first-out), processa a tarefa que está a mais tempo na fila;
- LWKR (Least Work Remaining), seleciona entre os produtos que estão na fila de espera aquele com o menor tempo restante de processamento.

As regras de despacho normalmente são fáceis de se implementar e podem ser baseadas na combinação de uma ou mais regras de processamento, ou em combinações específicas de um ambiente de produção (CÂNDIDO, 1997).

2.4.4.2 Algoritmos Genéticos

Os algoritmos genéticos (GA) foram desenvolvidos por John Holland na década de setenta (GOLDBERG, 1989).

Nos GA's as variáveis de uma função a ser otimizada são codificadas em um string, chamado cromossomo. Um cromossomo é composto por finitos genes, os quais podem ser representados por um alfabeto binário {0,1}, ou por um alfabeto de maior cardinalidade. Operadores de reprodução, cruzamento e mutação são aplicados aos cromossomos de uma população, garantindo a sobrevivência dos cromossomos mais aptos, e levando a uma melhoria da qualidade dos cromossomos (soluções) em gerações sucessivas. Exemplo da aplicação de GA em programação da produção são encontrados em MATTELD (1996).

2.4.4.3 Algoritmos de busca local

Algoritmos de busca local tentam melhorar continuamente uma solução gerada inicialmente por alguma heurística construtiva (CÂNDIDO, 1998a). Os algoritmos de busca local tem as seguintes elementos básicos:

- Esquema de representação: dependente do problema;

- Estrutura de vizinhança: a vizinhança de uma solução *s* é o conjunto soluções que podem ser obtidas pela aplicação de pequenas e predefinidas modificações em *s*, ou seja, pela aplicação de operadores de movimento em *s*;

- Estratégia de Controle: determina se a solução corrente vai ou não ser substituída por uma solução vizinha, e também qual será este vizinho. A estratégia de controle determina também quando interromper a busca. Exemplos: busca tabu, *simulated annealing*.

2.4.4.4 Busca Tabu

Os procedimentos de busca tabu vêm sendo usados largamente para resolver problemas de programação da produção. A busca Tabu é uma estratégia de controle para algoritmos de busca local.

Uma lista tabu impõe restrições para guiar o processo de busca, evitando ciclos e possibilitando a exploração de outras regiões do espaço de soluções. Na busca tabu escolhe-se o melhor vizinho, dentre os que não estão proibidos (aqueles que não estão na lista tabu).

A lista de movimentos proibidos é formada pelos movimentos opostos aos movimentos realizados mais recentemente. O movimento fica na lista tabu por um número limitado de passos. Logo após ele volta ao rol de movimentos permitidos. Assim a cada interação não são permitidos movimentos para um determinado subconjunto de vizinhos.

Os elementos básicos da busca tabu são: as estruturas de memória e o critério de aspiração. Existem dois tipos básicos de memória: a baseada na proximidade e a baseada na freqüência. Na memória baseada na proximidade um atributo permanece na lista tabu por um número t de movimentos após o atributo ter sido inserido na lista. A lista tabu normalmente opera com uma fila FIFO (First in – First out). O tamanho da lista tabu(t) pode ser estático ou

dinâmico. Na memória baseada na freqüência custos são atribuídos a certos movimentos de acordo com a freqüência com que o movimento ocorreu no passado. Este processo é fundamental para controlar a diversificação ou a intensificação do processo da busca.

O critério de aspiração permite um movimento proibido ser realizado se ele satisfazer a um critério. Exemplo de critério de aspiração: o vizinho é a melhor solução até o momento.

2.4.4.5 SIMULATED DE ANNEALING

Simulated annealing (SA) é uma técnica de busca com grande capacidade de sair de mínimos locais desenvolvida por KIRKPATRICK (1983).

SA tem uma analogia com as leis da termodinâmica, especialmente com o resfriamento de metais.

Com um resfriamento suficientemente lento do metal obtém-se um sólido com uma estrutura homogênea, com um mínimo de energia. Caso contrário, com um resfriamento muito rápido, obtém-se uma estrutura distante da estrutura de energia mínima.

Do ponto de vista da otimização combinatorial, SA é um algoritmo de busca local aleatória (JOHNSON & MCGEOCH 1997). SA permite movimentos montanha acima (para soluções piores). Estas mudanças são mais acentuadas quando o "metal" está em alta temperatura. O critério de aceitação de uma solução vizinha é explanado a seguir. Seja:

m_0 = solução corrente;
m_1 = solução vizinha;
c = temperatura atual (parâmetro de controle da busca);
F = função objetivo a ser minimizada;
P = probabilidade de aceitação da solução vizinha;

No algoritmo SA, em um problema de minimização, os movimentos em direção a uma solução melhor que a inicial são imediatamente aceitos isto é:

$P = 1$ se $F(m_1) < F(m_0)$

Já os movimentos em direção a soluções piores são aceitos com a seguinte probabilidade:

$P = \exp [- (F(m_1) - F(m_0))/c]$

Esta probabilidade diminui no decorrer da busca, a medida que a temperatura decresce. Em temperaturas elevadas (o parâmetro c com valores altos), a probalidade da solução vizinha pior ser aceita é alta. A medida que c decresce a probabilidade diminui (CÂNDIDO at al.,1998).

O algoritmo SA tem os seguintes parâmetros:

- Número de movimentos avaliados a cada temperatura (nsize);
- Temperaturas inicial e final (c_0 e c_f);
- Função de decremento da temperatura:
  Exemplo: Decremento exponencial: $c_{i+1} \leftarrow c_i * k$ onde $k<1$;

A estrutura de um algoritmo SA é desfrita a seguir:

Simulated_Annealing (c_0, c_f, nsize, k)

- Gerar uma solução inicial m_0 através de uma heurística construtiva
- $i \leftarrow 0$
- *count* $\leftarrow 0$

  Enquanto $c_i > c_f$ faça

    Enquanto *count* < *nsize* faça

      Selecione aleatoriamente uma solução m_1 na vizinhança de m_0

Calcule $F(m_1)$

Se { random(1) < exp {- ($F(m_1)$ - $F(m_0)$)/c_i } ou $F(m_1) < F(m_0)$ }então

$m_0 \leftarrow m_1$

count $\leftarrow$ *count* + 1

$c_{i+1} \leftarrow c_i * k$

count $\leftarrow$ 0

$i \leftarrow i + 1$

obs: Ramdon(1) é um número aleatório entre 0 e 1.

A qualidade e o tempo computacional em que a solução é encontrada depende do ajuste dos parâmetros c_o, c_f, k, e *nsize*.

2.5 Conclusão

Neste capítulo foi feita uma revisão das principais pesquisas realizadas sobre montagem de placas SMT. Também foi dada uma visão geral do problema do caixeiro viajante, da tecnologia de grupos e das técnicas ótimas e aproximadas utilizadas na resolução de problemas de *scheduling*.

3 Estrutura geral da metodologia proposta

3.1 Introdução

Neste capítulo são apresentadas a entrada dos dados do problema, dos parâmetros da máquina e a estrutura geral do sistema desenvolvido para abordar o problema.

A entrada dos dados do problema é feita através da planilha de cálculo Excel[2]. Estes dados foram divididos em duas tabelas:

1. os parâmetros da máquina na tabela PARAMETROS;
2. os dados das placas as serem produzidas na tabela ENTRADA.

A seguir apresentamos as principais tabelas de entrada e saída de dados e o fluxograma da metodologia adotada para solucionar o problema.

3.2 Entrada dos parâmetros e dados

A solução adotada neste trabalho procurou ser independente do tipo do fabricante de máquinas SMT (desde que de acordo com o descrito em **1.2.2**). Para tanto, a maioria das características da máquina foram parametrizadas. Esta parametrização facilitará a utilização do sistema desenvolvido neste trabalho em outras linhas.

A entrada dos parâmetros da máquina bem como os resultados e os gráficos gerados estão no formato de tabelas da planilha de cálculo Excel da Microsoft.

---

[2] Marca registrada da Microsoft.

O fato da solução ser desenvolvida em uma planilha de cálculo de uso genérico facilita o uso do aplicativo, uma vez que não é necessário nenhum software adicional para "rodar" o sistema.

3.2.1 Parâmetros da máquina

Na tabela PARAMETROS (Tabela 3) estão os seguintes dados da máquina de montagem SMT:

- Número máximo de alimentadores : número de alimentadores de 8mm que a máquina comporta;

- Tempo para retirar um alimentador: tempo necessário para retirar alimentador da mesa, tirar o carretel de componentes do alimentador e devolvê-lo e ao estoque;

- Tempo para colocar um alimentador : tempo para retirar o componente do estoque, inseri-lo no alimentador, e o tempo de colocar o alimentador na mesa;

- Tempo para trocar as mesas : tempo para desconectar as mesas da máquina mais o tempo para conectar as mesas preparadas com o próximo grupo de placas na máquina;

- Tempo para troca de programa : tempo necessário para carregar o programa da próxima placa a ser produzida;

- Tempo por componente; tempo médio de montagem de um componente pela máquina. Este valor servirá como referência para estimar o tempo de montagem de cada placa. Este tempo só será utilizado caso não seja informado pelo usuário o tempo médio de montagem da placa. Usualmente o valor informado pelo usuário é mais preciso;

Tabela 3. Parâmetros de entrada

			Setup Corrente	*Setup Extra*
Número max. De alimentadores	5	Pos1	R1	
Tempo para Retirar alimentador	0:06:00	Pos2	R2	
Tempo para colocar alimentador	0:05:00	Pos3	R7	
Tempo para trocar as mesas	0:15:00	Pos4	D1	
Tempo para troca de programa	0:05:00	Pos5	D5	
Tempo por componente	0:00:01	..		
Hora de Inicio	07:00	..		
Data Inicio	25/04/98	..		
Nome da linha	Linha 1	..		
Tempo de insersão da placa	0:00:30	..		
		..		
		Pos n		

- Data e hora a partir da qual pode-se iniciar a produção: esta data é útil para a confecção do gráfico de Gannt;

- Nome da linha: identificará a linha de produção na qual o processo está sendo executado;

- Tempo de inserção da placa: é o tempo que a máquina leva para inserir e posicionar a placa no local de composição;

- Colunas *Setup Corrente* e *Setup Extra*: indica a condição das mesas de composição no estado inicial. Informa o código dos componentes e a posição que estão nas mesas de composição corrente (mesas conectadas na máquina) e nas mesas extras (mesas que estão na área de preparação);

- Coluna Pos1, Pos2, ...: indica a posição do alimentador na mesa.

3.2.2 Dados das Placas

A tabela denominada ENTRADA (Tabela 4) fornece os dados das placas que serão produzidas. A tabela é composta dos seguintes itens:

- Núm. de placas: informa a quantidade de placas que será montada, ou seja, o tamanho do lote de cada placa;

Tabela 4. Dados da Tabela ENTRADA.

	Componentes									
Número de Placas	Tempo/Placa		Placa	R1	R2	R3	C1	C2	C3	D1	D2
5	0:00:55		a	2	0	5	0	2	0	0	0
6	0:00:45		b	3	0	0	1	0	0	9	0
3	0:00:40		c	0	8	0	0	1	1	0	0
2	0:00:42		d	0	0	0	0	3	0	0	9
5	0:00:00		e	4	0	4	0	0	1	0	5
7	0:00:50		f	0	8	3	0	0	0	0	1
3	0:00:35		g	2	0	0	0	0	0	1	0
4	0:00:55		h	4	0	0	5	0	6	0	8
5	0:01:10		I	5	0	0	8	7	0	14	0
12	0:00:55		j	6	4	0	0	0	0	0	9

- Tempo/Placa: tempo de montagem da placa fornecido pelo usuário. Caso este valor seja zero o sistema estimará o tempo de montagem multiplicando o número de componentes da placa pelo tempo médio por componente e somando o tempo de inserção da placa (dados fornecidos na tabela PARAMETROS);

- Placa: é a identificação da placa;

- Componentes (R1,R2,R3,C1,C2,C3,D1,D2): são os componentes das placas. Na tabela, zero indica a ausência do componente na placa. Um valor diferente de zero indica a quantidade de componentes na placa.

3.3 Fluxograma da Metodologia de Solução

Na **Erro! A origem da referência não foi encontrada.** está a estrutura geral da metodologia de solução. O primeiro passo será a formação de grupos de

placas. Após os grupos estarem formados será feito o seqüenciamento da produção. Dependendo do número de grupos formados será utilizado um algoritmo *branch and bound* ou um algoritmo de *simulated de annealing.*

Caso a seqüência sugerida não seja aceita pelo usuário, em função, por exemplo, de uma prioridade de produção, ele poderá entrar manualmente com a seqüência desejada e gerar um novo gráfico de Gannt. Neste caso, pode-se comparar os tempos da nova proposta com a solução originalmente apresentada.

Todo o programa foi escrito em VBA (Visual Basic for Aplications), (MCFREDIES, 1994).

3.4 Conclusão

Neste capítulo mostrou-se os dados de entrada do problema e a estrutura geral da metodologia proposta. Nos dois próximos capítulos serão descritos os aspectos relativos à formação de grupos e ao escalonamento da produção.

4 Formação de grupos

4.1 Introdução

O objetivo da formação de grupos é reduzir o número de *setups* necessário para a produção de uma determinado conjunto de placas. Neste capítulo será descrito o algoritmo desenvolvido para a formação de grupos de placas de circuito impresso.

A formação de grupo de uma forma genérica está descrita em ASKIN at al. (1993). SHTUB at al. (1991) abordou a formação de grupos em placas de circuito impresso. No presente trabalho foram utilizadas soluções análogas às apresentadas por SHTUB at al. (1991).

O algoritmo de formação de grupos visa determinar o menor número possível de grupos de placas. Na produção de placas de um mesmo grupo não há troca de *setups* das mesas de alimentadores, ou seja, todas as placas do grupo exigem apenas um *setup* das mesas. Assim, placas de um mesmo grupo têm um grande número de componentes em comum.

A formação de grupos foi dividida em:

- cálculo do coeficiente de similaridade entre as placas;
- algoritmo para formação de grupos.

4.2 Pré – processamento dos dados de entrada

As entradas do algoritmo de formação de grupos são:

- as PCI a serem montadas, fornecidas pela tabela ENTRADA, descrita no capítulo 3;

- os dados da máquina fornecidos pela tabela PARAMETROS.

Tabela 5. Matriz de Placas e Componentes, MBC

Número de Placas	Tempo / PLaca	Tempo do Lote	Número de Componentes	Número de Alimentadores		Nome da PLaca	R1	D2	C2	R2	R3	C1	C3	D1
5	0:00:55	0:04:35	9	3		A	1	0	1	0	1	0	0	0
6	0:00:45	0:04:30	13	3		B	1	0	0	0	0	1	0	1
3	0:00:40	0:02:00	10	3		C	0	0	1	1	0	0	1	0
2	0:00:44	0:01:24	12	2		D	0	1	1	0	0	0	0	0
5	0:00:44	0:03:40	14	4		E	1	1	0	0	1	0	1	0
7	0:00:50	0:05:50	12	3		F	0	1	0	1	1	0	0	0
3	0:00:35	0:01:45	3	2		G	1	0	0	0	0	0	0	1
4	0:00:55	0:03:40	23	4		H	1	1	0	0	0	1	1	0
5	0:01:10	0:05:50	34	4		I	1	0	1	0	0	1	0	1
12	0:00:55	0:11:00	19	3		J	1	1	0	1	0	0	0	0
						Σ	7	5	4	3	3	3	3	3

Para facilitar a visualização do processo será utilizado como exemplo ao longo deste capítulo um conjunto simples de dez placas, {a, b, c, d, e, f, g, h ,i ,j}, que serão montadas em uma máquina com capacidade para cinco alimentadores de 8mm, fc=5.

A matriz ENTRADA (Tabela 4) foi processada e os resultados foram armazenados na matriz de placas e componentes (MBC - Tabela 5), onde foram incluídas as seguintes informações:

- **Tempo /Placa:** foram verificadas quais as placas em que o usuário não forneceu o tempo de montagem, no caso a quinta placa. O sistema estimou o tempo de montagem fazendo cálculo: Tempo = (núm. Componentes * tempo de montagem do componente) + tempo de inserção = 14 * 1s + 30s=44s;

- **Tempo do lote:** é obtido multiplicando-se o número de placas pelo tempo de montagem de cada placa;

- **Número de alimentadores:** Foi inserida uma coluna com o número de alimentadores necessários para a montagem da placa. Esta coluna é calculada contando-se o número de componentes distintos da placa;
  Uma técnica para aumentar a velocidade de composição de PCI é colocar mais de um alimentador para os componentes muitos usados. Este procedimento não será permitido neste trabalho. Portanto o sistema alocará apenas um alimentador para cada tipo de componente;

- **Número de Componentes :** Foi inserida uma coluna com o número total de componentes da placa. O valor é o resultado da soma as linhas de componentes da tabela ENTRADA. Este número é utilizado para estimar o tempo de montagem da placa quando este tempo não for fornecido;

- **Componente mais usado:** Na última linha está a soma das colunas dos componentes, que indica qual componente é o mais usado no grupo;

As colunas dos componentes contém dados binários ao invés do número de componentes. MBCij=1 indica a presença do componente na placa e MBCij=0 a ausência. Esta transformação facilitará as operações E e OU no cálculo de similaridade necessário para a formação de grupos;

Os componentes foram ordenados decrescentemente da esquerda para direita. A chave de ordenação utilizada foi a soma dos componentes da coluna. Esta ordenação será utilizada para facilitar a alocação dos componentes nas mesas, de forma que os componentes mais utilizados fiquem mais próximos do cabeçote de composição, na primeira alocação.

Antes do cálculo do coeficiente de similaridade é feita uma comparação entre o número de alimentadores necessários para montar a placa e a capacidade de alimentadores da máquina, *fc*. Caso o número de alimentadores da placa exceda fc=5 a placa deverá ser dividida em duas placas. Por exemplo, caso haja uma placa X com oito componentes distintos, ela deve ser dividida em duas placas para efeito de

montagem: X_1 com cinco componentes e X_2 com três componentes[3]. O algoritmo de formação de grupos exclui a placa X da seleção e informa em quantos alimentadores a placa excedeu a capacidade da máquina.

4.3 Coeficiente de similaridade

Para o cálculo da similaridade de placas os vários coeficientes estudados foi escolhido o coeficiente de Jaccard (SOKAL, 1963), dentre os vários coeficientes estudados, pela sua simplicidade. Este coeficiente é utilizado nas mais variadas áreas do conhecimento humano para comparação de grupos. Existem na rede Internet vários exemplos da utilização do coeficiente de Jaccard que vão desde a classificação de tipos de *whiskies*[4] até a comparação de amostras biológicas. O coeficiente de Jaccard é definido por:

$$Srs = \frac{\eta_r \cap \eta_s}{\eta_r \cup \eta_s}$$

onde:

Srs = índice de similaridade entre r e s;

η_r = conjunto de componentes da placa r;

η_s = conjunto de componentes da placa s;

Abaixo está calculado o coeficiente de similaridade entre as placas a e b:

$$S_{ab} = \frac{\eta_a \cap \eta_b}{\eta_a \cup \eta_b} = \frac{1}{5} = 0{,}20$$

O valor do coeficiente de similaridade varia de zero a um, onde zero indica que não há nenhuma similaridade entre as placas e o valor um indica que as placas têm os

[3] Ver no capitulo 6 o item oportunidades de melhoria

[4] http://www.dcs.ed.ac.uk/~jhb/whisky/lapointe/text.html

mesmos componentes.

Calculando-se os valores dos coeficientes de similaridades entre as placas obtemos a matriz de similaridade do grupo, MS (Tabela 6).

Tabela 6. Matriz de Similaridade (MS)

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*I*	*j*	$\sum$
a	1,00	0,20	0,20	0,25	0,40	0,20	0,25	0,17	0,40	0,20	3,27
b	0,20	1,00	0,00	0,00	0,17	0,00	0,67	0,40	0,75	0,20	3,38
c	0,20	0,00	1,00	0,25	0,17	0,20	0,00	0,17	0,17	0,20	2,35
d	0,25	0,00	0,25	1,00	0,20	0,25	0,00	0,20	0,20	0,25	2,60
e	0,40	0,17	0,17	0,20	1,00	0,40	0,20	0,60	0,14	0,40	**3,68**
f	0,20	0,00	0,20	0,25	0,40	1,00	0,00	0,17	0,00	0,50	2,72
g	0,25	0,67	0,00	0,00	0,20	0,00	1,00	0,20	0,50	0,25	3,07
h	0,17	0,40	0,17	0,20	0,60	0,17	0,20	1,00	0,33	0,40	3,63
I	0,40	0,75	0,17	0,20	0,14	0,00	0,50	0,33	1,00	0,17	3,66
j	0,20	0,20	0,20	0,25	0,40	0,50	0,25	0,40	0,17	1,00	3,57

Na última coluna da matriz de similaridade está a soma dos índices de similaridade de cada placa. O maior valor desta coluna informará a placa mais similar ao conjunto de todas as placas. Esta coluna é uma inovação em relação ao trabalho de SHTUB at al. (1991).

No exemplo a placa "*e*" apresenta o maior grau de similaridade com as outras placas. A soma dos índices de similaridade da placa "*e* "com as outras placa é 3,68.

4.4 Algoritmo para formação de grupos

O algoritmo de formação de grupos apresentado neste trabalho utiliza uma heurística específica, (CANDIDO & BROCHONSKI, 1999). A partir da matriz de similaridade e da capacidade máxima de alimentadores da máquina (fc), o algoritmo agrupará as placas, tentando obter o menor número de grupos. Obviamente, o número de alimentadores utilizados em cada grupo não pode ultrapassar *fc*. O número de placas inicial é dez (nb=10).

Para facilitar a ordenação, os dados da matriz de similaridade foram organizados em

colunas (Tabela 7). Na primeira coluna estão os pares de placas, na segunda o índice de similaridade e na terceira a soma da linha. Ordena-se a matriz com duas chaves de ordenação decrescentes :

1. coeficiente de similaridade;
2. soma dos coeficientes.

Dentre todos os pares o par *l-b* é o par com o maior coeficiente de similaridade. As placas são 75% similares.

Tabela 7. Matriz Ordenada MS

Par	*Coeficiente de similaridade*	Σ
l - b	0,75	3,66
b - l	0,75	3,38
b - g	0,67	3,38
..	..	..
..	..	..
..	..	..
c - b	0,00	2,35
c - g	0,00	2,35

O número de linhas da matriz da Tabela 7 é calculado por núm_lin= (nb*nb)-nb.

A formação de grupos inicia-se com o par de placas { *l,b* } que está no topo da tabela ordenada MS. Calcula-se o número de alimentadores resultante da união das placas. Se o resultado união for superior a capacidade de alimentadores das mesas, *fc*, descarta-se este par e segue-se para o próximo par distinto, no caso (*b,g*). Como ($l \cup b$) = 4 é inferior a *fc* = *5*, forma-se o primeiro grupo G_1, composto pelas placas (*l,b*).

O número de placas é subtraído de um, ficando *nb*=9. Na matriz MBC são inseridas três colunas:

1. uma coluna onde será armazenado o número de placas do G_1 que é a soma do tamanho de lote de cada placa incluída no grupo, no caso (5+ 6) = 11 placas.

2. uma coluna onde é armazenado o tempo total de montagem do G_1 que é igual:

$$T\,G_1 = \sum_{i=1}^{\text{Núm. de Placas G1}} Tpi \quad \text{onde:}$$

T G_1 = Soma dos tempos de montagem das placas do G_1;

Tpi = tempo de montagem de cada placa;

3. uma coluna com o número de alimentadores necessário para a montagem do grupo;

Não é necessário calcular toda MS novamente, calcula-se apenas a similaridade entre o G_1, que está na primeira linha com o restante da placas. Ordena-se MS utilizando o coeficiente de similaridade como chave de ordenação decrescente (Tabela 8).

Tabela 8. Coeficientes de Similaridade Envolvendo o Par {l,b} Ordenados

Par	*Coeficiente de Similaridade*	Σ
(b - l) – g	0,50	2,91
(b - l) – a	0,40	2,91
(b - l) – h	0,33	2,91
(b - l) – d	0,20	2,91
(b - l) – c	0,17	2,91
(b - l) - j	0,17	2,91
(b - l) - e	0,14	2,91
(b - l) - f	0,00	2,91

Verifica-se a possibilidade de mais alguma placa da linha ser incluída no grupo, sem exceder o limite de alimentadores. Neste caso as placas *g* e *a* foram incluídas, (Tabela 9).

Tabela 9. Matriz de Similaridade (MS) com o Grupo G1

	b - l - g - a	*c*	*d*	*e*	*f*	*h*	*J*	
b - l - g - a	1,00	0,14	0,17	0,29	0,14	0,29	0,14	2,17
c	0,14	1,00	0,25	0,17	0,20	0,17	0,20	2,13
d	0,17	0,25	1,00	0,20	0,25	0,20	0,25	2,32
e	0,29	0,17	0,20	1,00	0,40	0,60	0,40	3,05
f	0,14	0,20	0,25	0,40	1,00	0,17	0,50	2,66
h	0,29	0,17	0,20	0,60	0,17	1,00	0,40	2,82
j	0,14	0,20	0,25	0,40	0,50	0,40	1,00	2,89

Copia-se G_1={*l,b,g,a*} para uma matriz denominada matriz de grupos concluídos MGC (Tabela 11).

Após concluir o grupo G_1 exclui-se a primeira linha e a primeira coluna de MS (Tabela 10), evitando-se assim o calculo de toda MS. Repete-se o processo até nb=0.

Tabela 10. MS Após Excluir a 1º Linha e a 1º Coluna

	c	*d*	*e*	*f*	*h*	*j*	
c	1,00	0,25	0,17	0,20	0,17	0,20	1,98
d	0,25	1,00	0,20	0,25	0,20	0,25	2,15
e	0,17	0,20	1,00	0,40	0,60	0,40	2,77
f	0,20	0,25	0,40	1,00	0,17	0,50	2,52
h	0,17	0,20	0,60	0,17	1,00	0,40	2,53
j	0,20	0,25	0,40	0,50	0,40	1,00	2,75

Os grupos formados neste exemplo estão na Tabela 11. 75% dos grupos utilizaram toda a capacidade das mesas alimentadoras.

Tabela 11. Conjunto Final de Grupos (MGC)

Grupos	*Placas*	*R1*	*D2*	*C2*	*R2*	*R3*	*C1*	*C3*	*D1*	*Tamanho do lote*	*Tempo de montagem*	*Nº. Alimentadores*
G1	b - l - g - a	1	0	1	0	1	1	0	1	19	00:16	5
G2	e - h	1	1	0	0	1	1	1	0	9	00:07	5
G3	f - j - d	1	1	1	1	1	0	0	0	21	00:18	5
G4	c	0	0	1	1	0	0	1	0	3	00:02	3

4.5 Conclusão

Neste capítulo foi apresentado um algoritmo de formação de grupos, que a partir de uma matriz de placas e componentes visa formar o menor número de grupos de placas, sem violar a capacidade de alimentadores.

5 Programação da produção com capacidade finita (*scheduling*)

5.1 Introdução

Neste capítulo serão tratados os algoritmos e as rotinas utilizadas para determinar a ordem em que os grupos de placas serão produzidos de forma a minimizar o tempo total de produção. A programação da produção (*scheduling*) será feita por dois algoritmos: um algoritmo de *branch and bound* (BB) que determina a seqüência ótima de grupos, e um algoritmo de *simulated annealing* (SA) que gera soluções sub-ótimas. O número de grupos determinará qual dos dois algoritmos será utilizado.

Para encontrar uma boa solução inicial para ser utilizada pelos algoritmos *Branch and bound* (como limite superior inicial) e SA (como ponto de partida) foi desenvolvida uma heurística construtiva.

Foi necessário também um algoritmo auxiliar (rotina monta) que decodifica uma seqüência de produção e calcula o tempo total de produção. As rotinas são explanadas a seguir.

5.2 Uma heurística construtiva para o problema de *scheduling* em máquinas SMT

A rotina denominada Heuri é uma heurística construtiva que visa a obter uma "boa" seqüência de produção. O *makespan* (vH) da seqüência sugerida pela rotina Heuri é utilizado pelo algoritmos *branch and bound (BB)* e *Simulated Annealing (SA)*. No algoritmo BB, este valor do *makespan* é utilizado como limite superior inicial. O quão mais próximo da solução ótima estiver este limite, mais eficiente será a busca. No algoritmo SA, a solução encontrada pela rotina Heuri é utilizada como solução inicial.

A heurística construtiva proposta é análoga à do vizinho mais próximo (*nearest neighbor*) para o problema do caixeiro viajante. Dada uma situação inicial de configuração das mesas de alimentadores, calcula-se o tempo de preparação de cada

grupo nas mesas extras (mesas na área de *setup*). Para tal, inicialmente tenta-se colocar os alimentadores nas mesas para montagem do grupo em questão. Caso não haja espaço para colocar todos os alimentadores do grupo, os alimentadores que estão nas mesas vão sendo retirados até que se libere o espaço necessário. Os tempos de cada atividade são computados.

O grupo com menor tempo de preparação, dentre os ainda não selecionados, é inserido na programação.

Calculam-se, então, os tempos de preparação das mesas correntes (mesas conectadas na máquina) e novamente escolhe-se o grupo com menor tempo de preparação, dentre os ainda não selecionados. Repete-se o processo, sempre levando em consideração os *setups* anteriores, até que todos os grupos sejam inseridos na programação.

Segundo reportado em COOK at al., (1998) em problemas TSP, da biblioteca TSPLIB[5] REINEIT, (1991), o custo do roteiro encontrado pelo algoritmo do vizinho mais próximo está em torno de 26% acima do custo da rota ótima. Segundo o mesmo autor é importante observar que o valor estimado de 26% é empírico, indicando apenas uma ordem de grandeza.

Na heurística construtiva são necessários os seguintes dados:

- a capacidade da máquina. Neste exemplo fc=5;
- os dados dos grupos, da condição inicial das mesas extras e correntes (Tabela 12).

Na Tabela 12 *Setup Extra* e *Setup Current* indicam a situação inicial dos pares de mesas extras e correntes respectivamente. Na condição inicial o par de mesas correntes está conectado na máquina.

[5] http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html

Tabela 12. Grupos e Mesas de alimentadores

Grupos	*R1*	*D2*	*C2*	*R2*	*R3*	*C1*	*C3*	*D1*
Setup Extra	0	0	0	0	0	0	0	0
Setup Current	1	0	0	1	0	0	0	0
G1	1	0	1	0	1	1	0	1
G2	1	1	0	0	1	1	1	0
G3	1	1	1	1	1	0	0	0
G4	0	0	1	1	0	0	1	0

Os tempos das operações utilizados são obtidos da tabela PARAMETROS (Tabela 3). As mesas extras são aos primeiras a serem preparadas. Para saber qual o primeiro grupo a ser preparado nas mesas extras deve-se calcular o tempo de preparação de todos os grupos nas mesas extras. Os resultados dos tempos de preparação estão na Tabela 13. Estes valores foram obtidos da seguinte forma:

1. é verificado a capacidade disponível (cap_disp) das mesas extras. A capacidade disponível é igual a capacidade de alimentadores da máquina menos o número de alimentadores que estão na mesas, cap_dis = fc – número de alimentadores nas mesas. No caso as mesas extras estão vazias, cap_disp= 5 – 0 = 5;

2. é verificado o número de alimentadores do G_1: $f\_G_1 = 5$;

3. a fim de determinar o número de alimentadores do G_1 que já estão presentes nas mesas extras é feita a interseção do G_1 com as mesas extras, neste caso $G_1 \cap$ mesas extras = 0;

4. determina-se a capacidade de alimentadores necessária (cap_nes):
   cap_nes = $f\_G_1 - (G_1 \cap$ mesas extras) = 5 – 0 = 5;

5. compara-se a capacidade necessária com a capacidade disponível. Caso a capacidade disponível não seja suficiente para comportar os alimentadores do grupo, faz-se a retirada do número necessários de alimentadores, para liberar o espaço necessário;

6. Determina-se o tempo total de preparação, obtido pela soma dos tempos de retirada e inserção de alimentadores. Da tabela Parâmetros o tempo para colocar cada alimentador é de 5 minutos. Como neste exemplo, serão necessários colocar cinco alimentadores, o tempo de preparação do G_1 é de 25 minutos.

Iniciando com as mesas extras calcula-se os tempos de preparação para todos os grupos e seleciona-se o com menor tempo de *setup*, no caso o grupo G_4 (Tabela 13). A seqüência terá inicio como grupo G_4. Os alimentadores do G_4 são então colocados nas mesas extras.

Enquanto a máquina está produzindo o G_4 com as extras, as mesas de alimentadores que estavam conectadas na máquina vão ser preparadas. Calcula-se o tempo de preparação dos três grupos restantes. Os grupos G1 e G3 tem os menores tempos de preparação, ambos com 21 minutos. Por ser o de menor ordem, o grupo G1 foi incluído na seqüência de montagem. A seqüência de produção está agora com os grupos {G4,G1}. Repete-se este procedimento até todos os grupos serem programados, obtendo-se a seqüência { G4, G1, G2, G3 }.

Note que para a inclusão do G3 nas mesas extras é necessário retirar das mesas extras o componente C3 e incluir R1, R3 e D2. Assim, o tempo de preparação das mesas extras com G3 é:

t_{pG3} = (6 x 1) + (3 x 5) =21 minutos.

Assim, a rotina Heurística obtém como solução a seqüência { G4, G1, G2, G3 }. Para calcular o tempo total de produção da seqüência deve-se levar em conta os tempos de espera, caso existam e os tempos de trocas de programa. A rotina Monta, descrita a seguir, calcula o tempo total de produção de uma seqüência.

5.3 Rotina para decodificação de solução

Dada uma seqüência de grupos a ser produzida é necessário um algoritmo para decodificar e fornecer o tempo total de produção.

A rotina Monta é a responsável pelo cálculo do *makespan* de uma seqüência considerando a situação inicial das mesas alimentadoras e os tempos de preparação e montagem de todos os grupos (BROCHONSKI & CANDIDO, 1998).

Para calcular o tempo total de produção a partir de uma seqüência devemos inserir os tempos:

- de troca de programas;
- de troca de mesas;
- de máquina parada aguardando a preparação das mesas alimentadores, caso existam.

Tabela 13. Seqüência das decisões em função do menor tempo

Condição Inicial

Grupos	*R1*	*R2*	*R3*	*C1*	*C2*	*C3*	*D1*	*D2*	*Nº. Feeders*	*Nº. Placas*
Setup Extra	0	0	0	0	0	0	0	0	0	
Setup Current	1	1	0	0	0	0	1	0	3	
G1	1	0	1	1	1	0	1	0	5	19
G2	1	0	1	1	0	1	0	1	5	9
G3	1	1	1	0	1	0	0	1	5	21
G4	0	1	0	0	1	1	0	0	3	3

Seqüência de grupos	**Tempo dos *SETUPs* (min)**						**Mesas após a inclusão do grupo**							
G4		*Setup Extra*	*Setup Current*	*G1*	*G2*	*G3*	*G4*	*Grupos*	*R1*	*R2*	*R3*	*C1*	*C2*	*C3*	*D1*	*D2*
	Setup Extra	0	0	25	25	25	15	**Setup Extra**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**
								Setup Current	1	1	0	0	0	0	1	0
G1		*Setup Extra*	*Setup Current*	*G1*	*G2*	*G3*	*G4*	*Grupos*	*R1*	*R2*	*R3*	*C1*	*C2*	*C3*	*D1*	*D2*
	Setup Current	0	0	21	32	21	x	Setup Extra	0	1	0	0	1	1	0	0
								Setup Current	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**0**
G3		*Setup Extra*	*Setup Current*	*G1*	*G2*	*G3*	*G4*	*Grupos*	*R1*	*R2*	*R3*	*C1*	*C2*	*C3*	*D1*	*D2*
	Setup Extra	0	0	x	32	21	x	**Setup Extra**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**1**
								Setup Current	1	0	1	1	1	0	1	0
G2		*Setup Extra*	*Setup Current*	*G1*	*G2*	*G3*	*G4*	*Grupos*	*R1*	*R2*	*R3*	*C1*	*C2*	*C3*	*D1*	*D2*
	Setup Current	0	0	x	22	x	x	Setup Extra	1	1	1	0	1	0	0	1
								Setup Current	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**1**

A rotina monta tem os seguintes parâmetros de entrada:

1. vetor n. Neste vetor está a seqüência de grupos a serem montados ex: n={ 4,1,3,2};

2. nível. É uma variável que determina até qual elemento do vetor deve ser calculado o tempo de produção. Esta variável será utilizada pelo algoritmo *branch and bound*, para calcular o tempo de produção de uma seqüência parcial. Para calcular o tempo de toda a seqüência basta fazer a variável nível igual à cardinalidade do vetor de entrada *n*.

O cálculo do tempo total de uma seqüência tem início com o *setup* das mesas extras. Para efeitos de exemplo será calculado o *makespan* da seqüência $n=\{G_4,G_1,G_3,G_2\}$. Calcula-se o tempo de preparação do primeiro grupo $n(1) = G_4$ nas mesas extras. O início da produção será: a data e hora de início da produção especificados na tabela Parametros, mais o tempo de troca das mesas somados com o tempo de preparação de G4. O término da produção de G_4 será o somatório dos tempos de montagem de cada grupo de placas com o tempo de troca de programa. A preparação de $n(2) = G_1$ que será feita nas mesas correntes começará quando forem retiradas as mesas correntes da máquina (Figura 8) . A Figura 8 está dividida em duas áreas: área de produção (montagem) onde estão as máquinas de montagem SMT, e área de preparação, onde as mesas são abastecidas e desabastecidas com componentes.

Enquanto a máquina esta produzindo o G_4, está acontecendo a preparação do grupo G_1. Duas situações poderão ocorrer:

1. $t_{mG4} > t_{pG1}$: tempo de montagem das placas do G_4 (t_{mG4}) maior que o tempo de preparação do G_1 (t_{pG1}). Neste caso não haverá tempo de espera da máquina. Quando a máquina terminar de montar o G_4 as mesas preparadas para montagem do G_1 estarão prontas para entrar em produção;

2. $t_{mG4} < t_{pG1}$. Neste caso, a máquina de montagem SMT ficará parada aguardando o término da preparação do G_1, o que é uma situação indesejável. Um dos principais objetivos deste trabalho é justamente minimizar este tempo de parada.

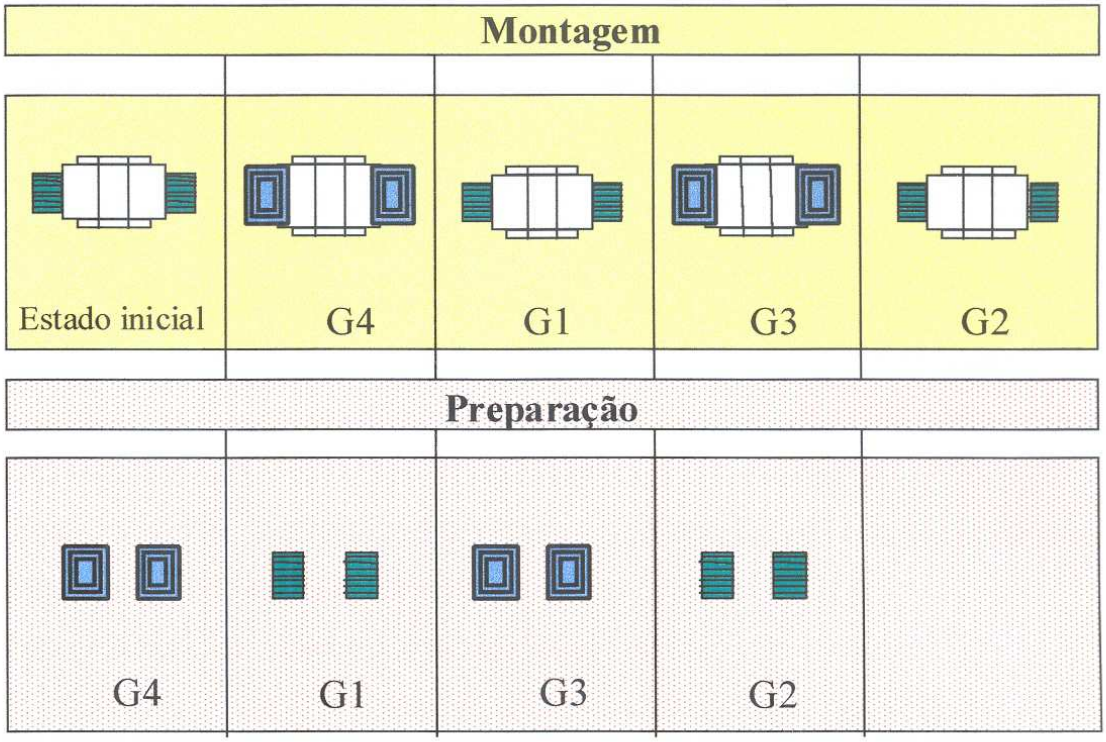


Figura 8. Seqüência de *setup* e montagem dos grupos

O G_3 será montado nas mesas extras. Neste momento as mesas extras estarão com o *setup* do G_1 e com os alimentadores que estavam inicialmente nas mesas, se existissem. Caso não haja espaço para os alimentadores do G_3 os alimentadores presentes nas mesas extras, não utilizados por G_3 , serão retirados até que se libere o espaço necessário para colocar todos os alimentadores de G_3. O cálculo dos tempos das operações envolvidas está descrito com mais detalhes no item 5.2.

Na primeira preparação das mesas extras e correntes são somados os tempos para retirar os componentes que não pertencem a nenhum grupo e que estão nas mesas na condição inicial. No caso foi somada à primeira preparação das mesas correntes 12 minutos referentes a retirada dos componentes R7 e D5.

Os resultados da rotina monta podem ser vistos na Tabela 14. Nesta tabela estão os seguintes dados dos grupos formados:

- Nome: nome do grupo;
- Mesa: as mesas com quais o grupo vai ser montado;
- Placas: placas que fazem parte do grupo;

- lote: total de placas que fazem parte do grupo, resultado do somatório do tamanho de cada lote de placas;
- Alimen.: quantidade de alimentadores de 8mm necessários para montagem do grupo;
- Partici.: número de tipos de placas participantes do grupo;
- Núm. Comp.: número total de componentes do grupo;
- Ordem de produção: indica a posição do grupo na seqüência de montagem;
- Tempo/Placa: indica o tempo necessário para montagem de cada placa, utilizado apenas quando se visualiza os dados da placa individualmente;
- Tempo Lote: tempo de montagem de todo o grupo de placas (não inclui tempos de troca de programa e troca de mesas);
- Início: data e horário do início da preparação do grupo;
- Início da produção: data e horário do início de montagem da primeira placa do grupo;
- Fim da produção: data e horário do término de montagem da última placa do grupo;

Tabela 14. Resultado da rotina Monta

Nome	*G4*		*G1*		*G3*		*G2*	
Mesas	*Extra*		*Corrente*		*Extra*		*Corrente*	
Placas	*c*		*b - l - g - a*		*f - j - d*		*e - h*	
lote	3		19		21		9	
Alimen.	3		5		5		5	
Partici.	1		4		3		2	
Núm. Comp.	11		59		43		37	
Grupo	4		1		3		2	
Ordem Produção	1		2		3		4	
Tempo lote	0:02:00		0:16:40		0:18:14		0:07:20	
Inicio	25/04	07:00	25/04	07:30	25/04	08:18	25/04	09:09
Inicio Produção	25/04	07:35	25/04	08:23	25/04	09:14	25/04	10:02
Fim Produção	25/04	07:37	25/04	08:54	25/04	09:42	25/04	10:15

0 tempo total de produção (makespan) é igual ao horário do fim de produção do último grupo menos o horário do início do primeiro grupo:

Makespan = grupo(n(nível)).fim – grupo(1).inicio = grupo(G1).fim – grupo(G_4).inicio = 10:15 – 07:00 = 03:15.

Adicionalmente foi construída a rotina Monta_setup, que trabalha da mesma forma que a rotina Monta, mas fornece mais informações (Tabela 15 e Tabela 16), tais como:

1 – a alocação dos alimentadores nas mesas;

2 – quantidade de componentes utilizada em cada grupo;

3 – os dados individuais de cada placa;

Tabela 15. Resultados Rotina Monta_setup com alocação de alimentadores

Nome	G4		G1		G3		G2	
Mesas	Extra		Corrente		Extra		Corrente	
Placas	c		b - l - g - a		f - j - d		e - h	
lote	3		19		21		9	
Alimen.	3		5		5		5	
Partici.	1		4		3		2	
Núm. Comp.	11		59		43		37	
Grupo	4		1		3		2	
Ordem Produção	1		2		3		4	
Tempo lote	0:02:00		0:16:40		0:18:14		0:07:20	
Inicio	25/04	07:00	25/04	07:30	25/04	08:18	25/04	09:09
Inicio Produção	25/04	07:35	25/04	08:23	25/04	09:14	25/04	10:02
Fim Produção	25/04	07:37	25/04	08:54	25/04	09:42	25/04	10:15
Pos.	Comp.	Quant.	Comp.	Quant.	Comp.	Quant.	Comp.	Quant.
1	C2	3	R1	36	C2	9	R1	24
2	R2	24	C2	27	R2	36	D2	39
3	C3	3	R3	15	R1	18	R3	12
4			C1	27	D2	57	C3	21
5			D1	72	R3	9	C1	15

O fornecimento destes dados extras demanda um tempo computacional maior. Portanto a rotina Monta_setup é chamada apenas na análise final dos resultados. Com os dados da Tabela 16 pode-se construir um gráfico de Gannt (Figura 9) onde se tem uma visão geral dos procedimentos de preparação das mesas extras e correntes e da montagem das placas.

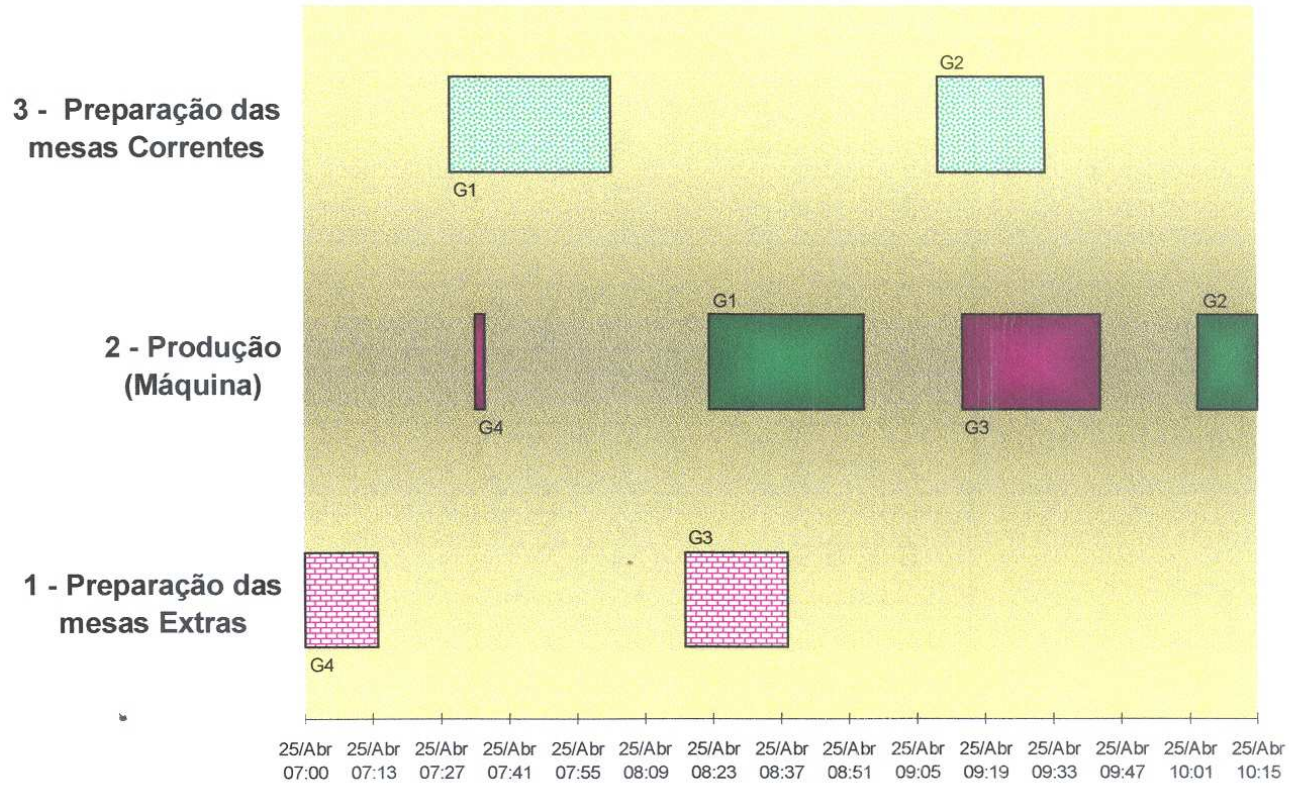


Figura 9. Visão da Programação da Produção dos grupos

Tabela 16. Placas com Alocação de Alimentadores e horários

Nome	***a***		***b***		***c***		***d***		***e***		***f***		***g***		***h***		***I***		***j***	
Mesas	*Cor.*		*Cor.*		*Ext.*		*Ext.*		*Cor.*		*Ext.*		*Cor.*		*Cor.*		*Cor.*		*Ext.*	
Placas	*a*		*b*		*c*		*d*		*e*		*f*		*g*		*h*		*I*		*j*	
lote	5		6		3		2		5		7		3		4		5		12	
Alimen.	3		3		3		2		4		3		2		4		4		3	
Partici.	-		-		-		-		-		-		-		-		-		-	
Núm. Comp.	9		13		10		12		14		12		3		23		34		19	
Grupo	1		1		4		3		2		3		1		2		1		3	
Ordem Produção	4		2		1		3		1		1		3		2		1		2	
Tempo / Placa	0:00:55		0:00:45		0:00:40		0:00:42		0:00:44		0:00:50		0:00:35		0:00:55		0:01:10		0:00:55	
Tempo lote	0:04:35		0:04:30		0:02:00		0:01:24		0:03:40		0:05:50		0:01:45		0:03:40		0:05:50		0:11:00	
Inicio	25/04	07:30	25/04	07:30	25/04	07:00	25/04	08:18	25/04	09:09	25/04	08:18	25/04	07:30	25/04	09:09	25/04	07:30	25/04	08:18
Inicio Produção	25/04	08:50	25/04	08:33	25/04	07:35	25/04	09:41	25/04	10:02	25/04	09:14	25/04	08:43	25/04	10:11	25/04	08:23	25/04	09:25
Fim Produção	25/04	08:54	25/04	08:38	25/04	07:37	25/04	09:42	25/04	10:06	25/04	09:20	25/04	08:45	25/04	10:15	25/04	08:28	25/04	09:36
Pos.	Comp.	Quant.	Comp.	Quant.	Comp.	Quant.	Comp.	Quant.	Comp.	Quant.	Comp.	Quant.	Comp.	Quant.	Comp.	Quant.	Comp.	Quant.	Comp.	Quant.
1	R1	2	R1	3	C2	1	C2	3	R1	4			R1	2	R1	4	R1	5		
2	C2	2			R2	8			D2	5	R2	8			D2	8	C2	7	R2	4
3	R3	5			C3	1			R3	4							D1	14	R1	6
4			D1	9			D2	9	C3	1	D2	1	D1	1	C3	6			D2	9
5			C1	1							R3	3			C1	5	C1	8		

5.4 Algoritmo Branch and Bound

O problema da programação da produção dos grupos formados é um problema NP-completo. O algoritmo *branch and bound* determina uma solução ótima, percorrendo de maneira inteligente uma árvore de busca. Em um problema com quatro grupos, o espaço de busca envolve vinte e quatro possibilidades (Figura 10). A estrutura de árvore formada tem 4 níveis. Os grupos foram nomeados de um a quatro {1,2,3,4}. O custo envolvido é a soma dos tempos de montagem e de preparação de um grupo para o outro. O tempo total de preparação e montagem para a seqüência {1,2,3,4} é $c_{1,4} = (t_{mG1} + t_{mG2} + t_{mG3} +$

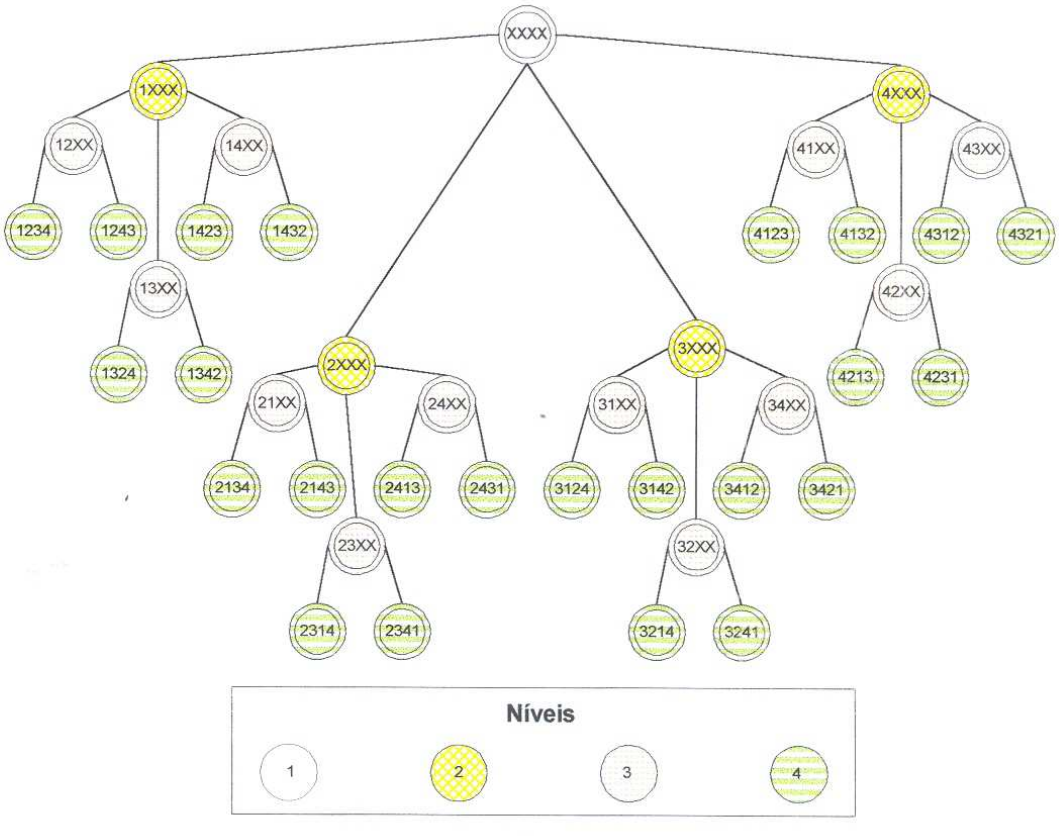


Figura 10. Estrutura de árvore

$t_{mG4} + t_{G1 \Rightarrow G2} + t_{G1G2 \Rightarrow G3} + t_{G1G2G3 \Rightarrow G4})$, onde:

t_{mGi} = tempo de montagem do grupo i

$t_{G1G2..Gi \Rightarrow Gi+1}$ = tempo de *setup* do grupo Gi+1 após a sequencia G1G2....Gi.

Caso as vinte e quatro soluções fossem pesquisadas teria-se uma busca exaustiva. O algoritmo *branch and bound* ramifica os grupos passo a passo, realizando, no caso proposto, uma busca em profundidade. A cada ramificação, é calculado um limite inferior (LB) para o ramo. Este LB é comparado com um limite superior vigente (vH). O limite superior inicial correspondente ao *makespan* da solução obtida pela rotina Heuri. O LB será o valor de cij, tempo real de montagem, mais t_min, limite inferior estimado de montagem dos grupos que faltam (LB = c_{ij} + t_min). Se o LB for superior a vH a busca é interrompida no ramo corrente e continua em outro ramo da árvore. Caso contrário a busca continua no mesmo ramo. Na comparação as seguintes situações podem ocorrer:

- **LB >= vH:** a rotina interrompe a busca nos ramos inferiores e continua a pesquisa no próximo ramo não computado;

- **LB < vH e nível < n_de_grupos :** ramifica mais um nível;

- **LB < vH e nível = n_de_grupos:** a rotina encontrou uma solução melhor que a heurística, o caminho é armazenado em uma variável denominada melhor_caminho e vH = c_{ij}.

5.4.1 Cálculo do LB

O valor de t_min é calculado pela rotina conta_alimetadores. A rotina recebe um valor indicando até que ponto a árvore foi ramificada. Então calcula:

- t_setup: tempo de preparação dos grupos que restam;
- t_placas: o tempo de montagem das placas ainda não montadas;

da seguinte forma:

t_placas = n_placas * par.programa

```
                +    grupos_restantes * par.troca
                +    (lote - lote_parcial)

t_setup =            n_alim * par.coloca
                +    n_and * par.coloca
                +    restam * par.retira _
                +    grupos_restantes * par.troca _
                +    grupos_restantes * par.programa _
                +    menor_lote
```

onde:

- n_placas: número de placas a serem montadas;
- par.programa : tempo para troca de programa entre placas;
- grupos_restantes : número total de grupos menos número de grupos já montados;
- par.troca : tempo de troca das mesas;
- lote : tempo de montagem de todas as placas;
- lote_parcial : tempo das placas já montadas;
- n_alim : número de alimentadores que faltam ser colocados;
- par.coloca : tempo para colocar um alimentador;
- restam: quantidade de alimentadores a serem retirados. Se n_alim for maior que a capacidade da máquina será necessário retirar alimentadores da mesas: n_alim > fc restam= n_alim – fc e restam = 0 caso contrário,
- n_and : número de componentes que serão necessários serem montados nas mesas extras e correntes. Indica que será necessário um alimentador nas mesas extras e outro na corrente;
- menor_lote : tempo de montagem do menor lote ainda não montado, último grupo a ser montado (Figura 9);

O valor de t_min será o maior valor entre t_setup e t_placas. O cálculo do LB depende de t_min que demanda grande tempo computacional.

Os valores de c_{ij}, t_min e LB podem ser vistos na Tabela 17. O ramo que inicia com G_3 não foi pesquisado. Nos ramos G_1, G_2, G_4 a pesquisa foi parcial. Se LB for menor que vH o algoritmo ramifica mais um nível. Caso contrário a busca é encerrada neste ramo e continua no próximo.

A solução ótima para este problema encontra-se no seqüência $\{G_1, G_3, G_2, G_4\}$. O tempo total de produção é 2:55:34 horas.

Tabela 17. Resultados da busca utilizando o algoritmo *Branch and Bound*

Caminho	vH	cij	t_min	LB	árvore
G1	3:15:14	1:16:40	1:57:34	3:14:14	Ramifica
G1,G2	3:15:14	1:56:20	1:15:14	3:11:34	Ramifica
G1,G2,G3	3:15:14	2:49:14	0:32:00	3:21:14	Corta
G1,G2,G4	3:15:14	2:23:00	1:03:14	3:26:14	Corta
G1,G3	3:15:14	2:01:14	1:09:20	3:10:34	Ramifica
G1,G3,G2	3:15:14	2:33:34	0:32:00	3:05:34	Ramifica
G1,G3,G2,G4	3:15:14	***2:55:34**	0:00:00	2:55:34	
G1,G3,G4	**2:55:34**	2:23:14	0:47:20	3:10:34	Corta
G1,G4	2:55:34	1:24:00	1:30:34	2:54:34	Ramifica
G1,G4,G2	2:55:34	2:11:20	0:58:14	3:09:34	Corta
G1,G4,G3	2:55:34	2:27:14	0:47:20	3:14:34	Corta
G2	2:55:34	0:57:20	1:51:54	2:49:14	Ramifica
G2,G1	2:55:34	2:04:40	1:15:14	3:19:54	Corta
G2,G3	2:55:34	2:01:14	1:03:40	3:04:54	Corta
G2,G4	2:55:34	1:24:00	1:24:54	2:48:54	Ramifica
G2,G4,G1	2:55:34	2:30:40	0:58:14	3:28:54	Corta
G2,G4,G3	2:55:34	2:27:14	0:41:40	3:08:54	Corta
G3	2:55:34	1:13:14	1:46:00	2:59:14	Corta
G4	2:55:34	0:37:00	2:07:14	2:44:14	Ramifica
G4,G1	2:55:34	1:54:40	1:30:34	3:25:14	Corta
G4,G2	2:55:34	1:46:20	1:24:54	3:11:14	Corta
G4,G3	2:55:34	1:51:14	1:19:00	3:10:14	Corta

* encontrada melhor solução

5.5 Simulated Annealing

Simulated Annealing (SA) pertence à classe de algoritmos de busca local. É um algoritmo que está sendo utilizado em uma gama variada de problemas. SA é de fácil aplicação e geralmente apresenta bons resultados. Segundo AARTS & LENSTRA (1998) SA tem uma componente estocástica que facilita a análise teórica de sua convergência, o que o fez muito popular entre os matemáticos.

Uma característica importante do algoritmo é sua capacidade de aceitar movimentos que não melhoram a função custo. Esta característica confere ao algoritmo a capacidade de sair de mínimos locais. Nos seus anos de existência SA já foi utilizado em uma gama grande de aplicações práticas e de problemas teóricos .

5.5.1 A analogia com a física

A origem da técnica SA vem de uma analogia com a o processo de resfriamento de metais previamente fundidos. O processo foi descrito por KIRKPATRICK at al. (1983). Inicialmente a temperatura do metal é aumentada até o metal tornar-se líquido. O metal fundido é então cuidadosamente resfriado até atingir o estado sólido. Se o resfriamento for feito de maneira suficientemente lenta, é estrutura cristalina do sólido corresponderá à de mínima energia.

Este processo pode ser modelado por computador. Uma das primeiras técnicas foi introduzida por Metropolis et al. (1953), que propôs um algoritmo para simular o resfriamento de um metal. Neste algoritmo a probabilidade do sólido passar de um estado de energia F(D) para um estado de energia F(D') é dado por:

$P = 1$, se $F(D') < F(D)$

$P = \text{Exp}\{ (F(D') - F(D)) / c \}$, caso contrário

O parâmetro de controle *c* corresponde a temperatura no modelo físico*. O valor de *c* é gradualmente decrescido durante o processo de busca. Esta regra de aceitação é conhecida por critério de Metropolis[6].

Mesmo em baixas temperatura, existe a chance de o sistema estar em um estado de alta energia, embora esta probabilidade seja baixa. Esta característica é utilizada no algoritmo SA. Em temperaturas altas a probabilidade do sistema sair de um mínimo local é maior que em baixas temperaturas. Em outras palavras o sistema pode ir montanha acima bem como montanha abaixo, mais em baixas temperaturas é menos provável que o sistema vá montanha acima (PRESS at al., 1990).

No início do processo (altas temperaturas) a probabilidade de aceitar soluções piores é alta. A probabilidade diminui durante o processo de resfriamento, até que o processo congele e soluções piores não são mais aceitas. É importante frisar que soluções melhores são aceitas com probabilidade 1 durante todo o processo de busca (CÂNDIDO at al., 1998).

Usando a teoria das cadeias finitas de Markov, Aarts prova que assintoticamente SA encontra solução ótima com probabilidade 1. (AARTS & LENSTRA, 1998). Mas para tanto é necessário um número muito grande de transições, maior que o espaço de busca. Para garantir uma solução ótima com SA para o problema do caixeiro viajante a complexidade computacional seria:

$$O(n^{n^{2n-1}})$$

(AARTS & LENSTRA, 1998). Portanto para garantir a solução ótima para o TSP utilizando SA gastaria muito mais tempo que uma busca exaustiva cuja

---

[6] No algoritmo Metrópolis c = t.k, onde t= temperatura, e K= constante de Boltzman

complexidade é fatorial. Trabalhando com um número finito de soluções não é possível garantir uma solução ótima, mas é possível encontrar uma solução aproximada boa com um tempo computacional razoável.

Utilizando SA com um número finito de transições, para que o algoritmo chegue a uma solução boa em um tempo razoável, é necessário o ajuste dos paramentos de resfriamento (*cooling schedule*). Estes parâmetros vão governar a convergência do algoritmo, conforme descrito por LAARHOVEN at al. (1992). São eles:

- N_over :o número de transições a cada temperatura (comprimento da cadeia de Markov);

- c_0 : o valor inicial da temperatura;

- c_f :o valor final da temperatura, ou outra regra de terminação;

- k : o valor de decremento da temperatura (no caso do resfriamento exponencial);

Estes parâmetros de controle podem ser estáticos ou dinâmicos . Os estáticos são fixos, não se alteram durante a execução do algoritmo. Os dinâmicos se alteram adaptando-se a execução do algoritmo com o objetivo de melhorar a performance do algoritmo. Neste trabalho foram utilizados apenas parâmetros estáticos.

5.5.2 Parâmetros de Controle e Estruturas do Algoritmo SA

Abaixo estão descritos os parâmetros de controle e as estruturas necessárias para aplicação do SA:

1. Esquema de representação: foi utilizado permutação de grupos, como por exemplo {G1, G2, G3, G4};

2. Procedimento de decodificação: dada uma solução codificada *s*, o procedimento de decodificação utilizado foi a rotina MONTA, descrita no item 5.2.

3. Estrutura de Vizinhança: foram realizados testes com três estruturas de vizinhança:

   - **troca simples :** Dada uma seqüência de grupos (s), a vizinhança N(s) de (s) é o conjunto de soluções obtidas pela troca de dois grupos consecutivos. Por exemplo: com (s) = (G1, G2, G3, G4), N(s)={ (G2, G1, G3, G4), (G1, G3, G2, G4), (G1, G2, G4, G3) ,(G4, G2, G3, G1) } ;

   - **2-opt :** A estrutura de vizinhança 2-opt ocasiona uma modificação simples de (s) fazendo trocas de percurso. O algoritmo foi proposto em 1958 por Croes (AARTS & LENSTRA, 1998). Dado um problema com n nós, onde cada nó representa um grupo, e os arcos conectando os grupos representam o tempo de preparação entre os grupos (Figura 11), a vizinhança 2-opt seleciona dois arcos aleatoriamente e desfaz as ligações entre os grupos. Então os grupos são reconectados de uma outra forma, gerando assim um novo percurso, como mostra o exemplo da Figura 11.

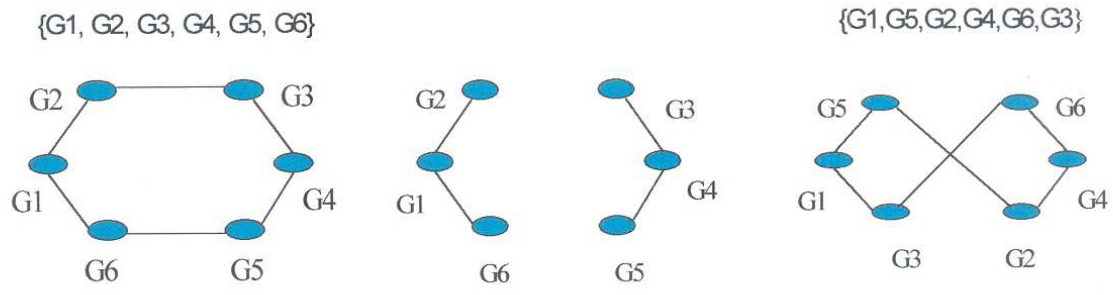


Figura 11. Estrutura de Vizinhança 2-opt

- **3-opt.** A estrutura 3-opt é similar a 2-opt. No percurso ao invés de 2 arcos são selecionados aleatoriamente três arcos. Os arcos são reconectados novamente de forma diferente da original (Figura 12) . 3-opt será usado quando o número de grupos for igual ou superior a seis.

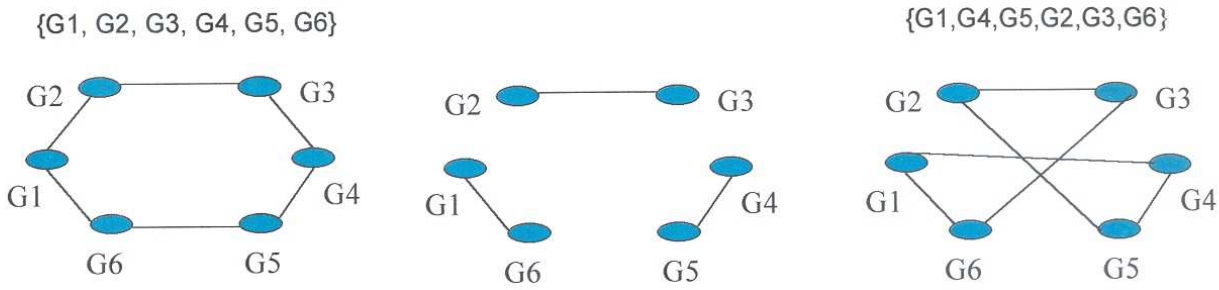


Figura 12. Vizinhança 3-OPT.

4. Parâmetros de controle do resfriamento ("*cooling schedule*"):

- Número de vizinhos a serem testados a cada temperatura (nover). **nover** é igual ao número de grupos
- Temperatura inicial (c_0):

  c_0 = (*Δ 10 makespan*) / Ln(1 / percentual aceitação))

  onde:

  - *Δ 10 makespan* : diferença entre o máximo e o mínimo *makespan* de dez seqüências geradas aleatoriamente;

  - percentual de aceitação : probabilidade de aceitação de um movimento que leva da pior à melhor solução entre as 10 geradas aleatoriamente. O percentual adotado foi de 70%.

- Decréscimo da temperatura. Foi utilizado um decréscimo exponencial da temperatura:

  $c_j = c_0 * k^j$ onde c_j é o valor da i-ésima temperatura;

  k = 0,995 é a taxa de resfriamento, determinada empiricamente;

- mecanismo de parada: o algoritmo SA termina a execução quando uma das seguintes situações ocorrer:
- quando c_j for menor que a temperatura final *c_f = 0,002;
- após 350* mudanças de temperatura sem alterações na solução corrente.

* valores determinados empiricamente;

Uma visão global do algoritmo SA está na Figura 13.

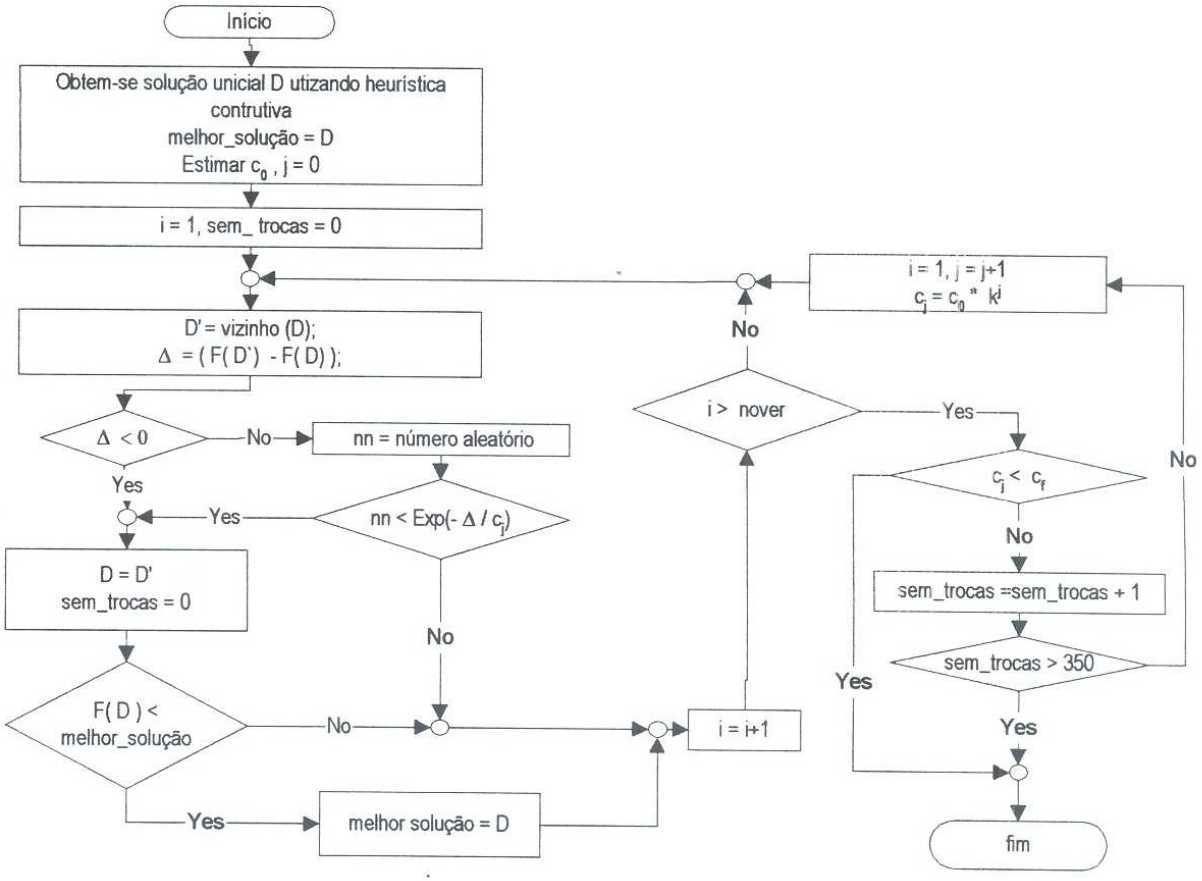


Figura 13. Fluxograma *Simulated Annealing*

5.6 Conclusão

Neste capítulo foram apresentadas todas as rotinas utilizadas no escalonamento da produção. Uma boa solução inicial foi obtida utilizando um algoritmo específico análogo ao do vizinho mais próximo para o problema do caixeiro viajante. A rotina Monta foi desenvolvida para decodificar uma seqüência de produção, levando em conta todos *setups* precedentes e calculando o tempo total de produção. Além do *makespan* a rotina Monta configura as mesas alimentadoras e apresenta a seqüência de produção em um gráfico de *Gannt*, onde se tem uma visão global do processo. Um algoritmo *Branch* and *Bound* foi desenvolvido para tratar instâncias menores do problema. Também um Algoritmo *Simulated Annealing* com suas estruturas e parâmetros foi explorado. No próximo capítulo estão descritos os resultados práticos das rotinas desenvolvidas neste capítulo.

6 Resultados

6.1 Introdução

Neste capítulo são apresentados os resultados da formação de grupos e os da programação da produção, envolvendo os algoritmos *branch and bound* e *simulated annealing.* Foram feitos testes com SA com três estruturas de vizinhança: troca simples, 2-opt e 3-opt. Os testes foram realizados em 24 problemas distintos. Nestes testes foram envolvidos 520 tipos diferentes de placas.

6.2 Descrição dos problemas utilizados nos testes

Para verificar o desempenho dos algoritmos foram elaboradas quatro baterias de testes (Tabela 18) denominadas seleção I, II, III e IV. Cada bateria é composta de 130 tipos diferentes de placas, divididas em seis problemas com cinco, dez, quinze, vinte, trinta e cinqüenta tipos de placas respectivamente. Assim, foi utilizado um total de vinte e quatro problemas (problemas *a* a *z*).

As quatro seleções se diferem em relação ao número de componentes por placa ou à diversidade de componentes da seleção. Os testes foram realizados em um Pentium[7] 233MHz.

6.3 Resultados Formação de Grupos

Para verificar o desempenho do algoritmo de formação de grupos foram utilizadas as placas da Tabela 18. Em todos os testes o número de alimentadores livres na máquina em relação à média de componentes por

[7] Pentium é marca registrada da empresa Intel

Tabela 18. Testes Realizados

Seleção	Problema	Número de placas	fc (capacidade de alimentadores)	Número de componentes médio por placa	Diversidade (componentes distintos na seleção)
I	a	5	50	42	80
	b	10			
	c	15			
	d	20			
	e	30			
	f	50			
II	g	5	18	15	20
	h	10			
	i	15			
	j	20			
	l	30			
	m	50			
III	n	5	80	67	141
	o	10			
	p	15			
	q	20			
	r	30			
	s	50			
IV	t	5	50	42	200
	u	10			
	v	15			
	w	20			
	x	30			
	y	50			

placa é de aproximadamente 20%.

Os tempos computacionais da formação de grupos e o número de grupos formado em cada problema estão na Tabela 19.

A relação entre o número de placas e o número de grupos pode ser visualizada na Figura 14. Na figura cada raio do circulo representa um problema onde as marcas redondas (vermelhas) representam o número de preparações iniciais necessárias considerando um *setup* para cada tipo de placa e as marcas com o sinal "+" o número de *setups* após a formação de grupos (número de grupos). Por exemplo, no problema *m* o número de trocas de *setups* caiu de 50 para apenas 11, com o agrupamento de placas. Só na redução do número de *setups* já houve um ganho significativo de tempo.

Tabela 19. Tempos Para de Formação de Grupos.

Seleção											
I			II			III			IV		
Problema (Núm. Placas)	Núm. Grupos	Tempo computacional mm:ss	Problema (Núm. Placas)	Núm. Grupos	Tempo computacional mm:ss	Problema (Núm. Placas)	Núm. Grupos	Tempo computacional mm:ss	Problema (Núm. Placas)	Núm. Grupos	Tempo computacional mm:ss
a (5)	3	00:03	g (5)	4	00:02	n (5)	3	00:04	t (5)	5	00:05
b (10)	6	00:08	h (10)	6	00:05	o (10)	5	00:11	u (10)	7	00:14
c (15)	9	00:15	i (15)	5	00:10	p (15)	9	00:21	v (15)	12	00:34
d (20)	11	00:26	j (20)	9	00:17	q (20)	11	00:36	w (20)	17	00:56
e (30)	17	01:02	l (30)	7	00:44	r (30)	14	01:25	x (30)	16	01:45
f (50)	31	03:21	m (50)	11	02:41	S (50)	24	04:24	y (50)	17	05:28

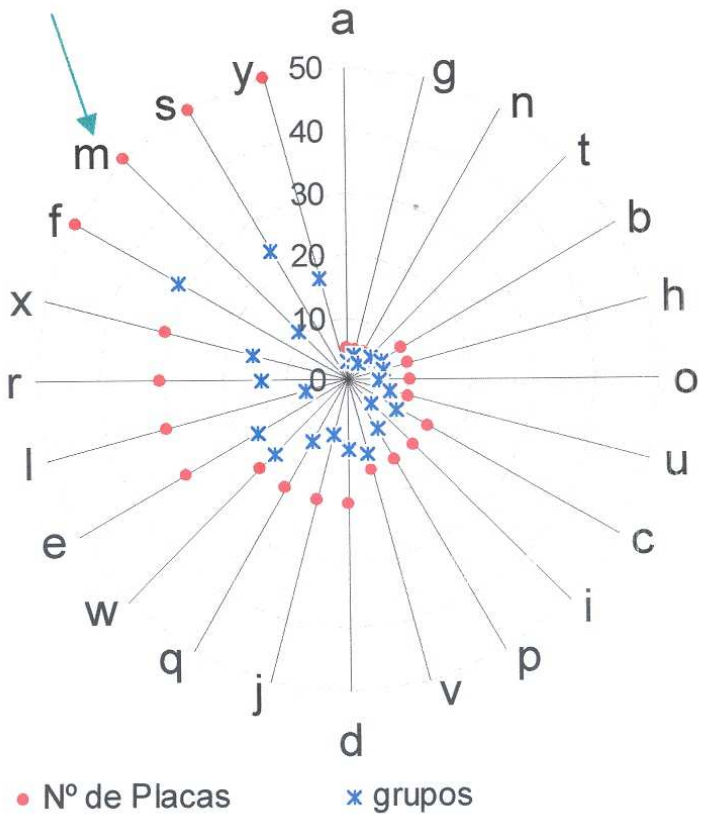


Figura 14. Problema, Número de *setups* iniciais, Número de *setups* após a formação de Grupos

Os resultados dos tempos por placa da Tabela 19 foram colocados no gráfico da Figura 15. O tempo computacional para formação de grupos sofre influência de três fatores: o número de placas do problema, o número médio de componentes por placa, e o número de componentes distintos no problema (diversidade de componentes).

Vemos no gráfico da Figura 15 que o fator que determina a forma da curva é o número de placas. Em todas as seleções o aspecto da curva é o mesmo. Para até 30 placas o tempo de formação de grupos é inferior a dois minutos mostrando que o algoritmo atende perfeitamente as necessidades para um horizonte de planejamento de curto prazo.

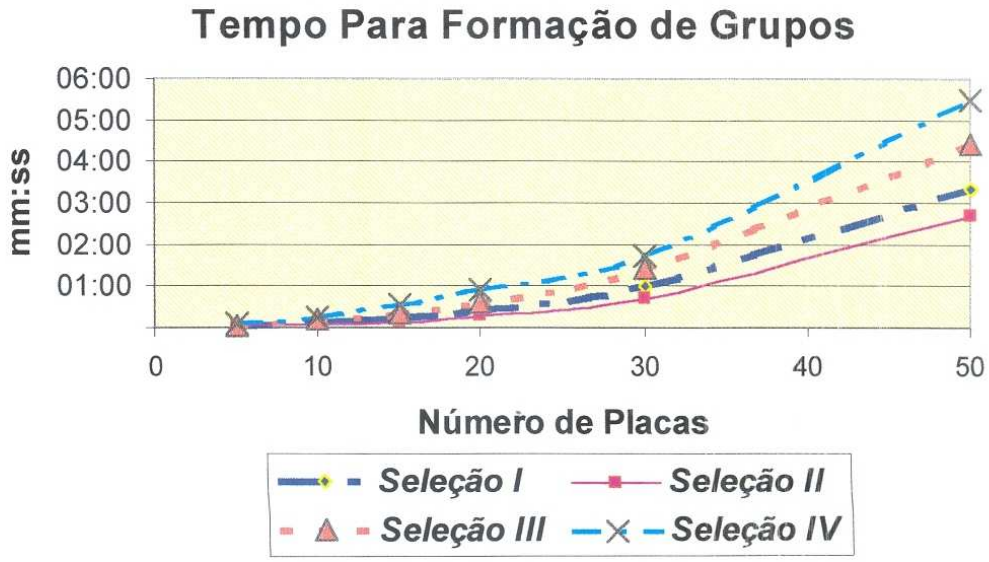


Figura 15. Tempo computacional Resultados comparativos da Formação de Grupos

Para se ter uma visão comparativa dos resultados da formação de grupos tomou-se a situação original do problema, ou seja a montagem de placas sem a formação de grupos. Os tempos totais de produção (*makespan*) dos problemas utilizados nos testes (Tabela 18), estão na Tabela 20. Abaixo está a descrição dos dados da tabela:

Tabela 20. Makespan Placas e Placas Agrupadas

Problema	*Núm de Placas*	*Núm. de Grupos*	*Média 30 Placas**	*Heurística Placas**	*Média 30 Grupos**	*Heurística Grupos**	*Ganho %*
a	5	3	18:32	17:25	12:51	13:43	74%
n	5	3	27:53	26:55	20:33	20:45	74%
g	5	4	6:31	6:04	5:28	5:30	85%
i	15	5	15:06	12:35	7:51	7:50	52%
o	10	5	62:45	45:16	34:33	33:20	53%
t	5	5	31:56	29:55	31:49	29:44	93%
b	10	6	34:09	26:02	21:57	19:46	58%
h	10	6	11:05	9:44	6:44	6:33	59%
l	30	7	27:54	21:04	10:49	11:18	41%
u	10	7	59:51	57:55	44:15	42:06	70%
c	15	9	49:58	36:45	33:38	28:00	56%
j	20	9	18:58	16:00	10:32	10:38	56%
p	15	9	86:36	57:43	55:16	47:52	55%
d	20	11	67:50	43:54	38:58	33:48	50%
m	50	11	45:39	35:33	17:03	18:01	39%
q	20	11	117:27	97:19	67:51	61:55	53%
v	15	12	86:38	52:54	73:02	51:03	59%
r	30	14	176:13	126:19	89:17	73:22	42%
x	30	16	116:35	78:44	87:43	51:31	44%
e	30	17	102:37	64:41	60:04	51:32	50%
w	20	17	83:09	55:25	87:38	53:24	64%
y	50	17	152:19	98:19	96:04	58:08	38%
s	50	24	291:58	182:39	149:31	125:36	43%
f	50	31	165:42	109:56	110:50	84:02	51%

* tempo em hh:mm

- Problema: problema especificado na Tabela 18;
- Núm de Placas : número de placas do problema;
- Núm. de Grupos: número de grupos formados após a aplicação do algoritmo de formação de grupo descrito capítulo 4;
- Média 30 Placas: média do *makespan* de trinta soluções geradas aleatoriamente da montagem das placas, sem a aplicação do algoritmo de formação de grupos, levando em conta apenas o *setup* imediatamente anterior;

- Heurística Placas: *makespan* da solução gerada pela heurística[8] construtiva descrita no capítulo 5.2, considerando cada placa individualmente, ou seja, sem a prévia formação de grupos;
- Média 30 Grupos: média do *makespan* de trinta seqüências aleatórias de grupos de placas;

Heurística Grupos : *makespan* da solução gerada pela heurística. Construtiva descrita no capítulo **5.2**, aplicada aos grupos de placas;

Os ganhos percentuais entre os tempos de produção das placas, sem a prévia formação de grupos e os tempos de produção dos grupos escalonados pela rotina Heurística variam de 38% a 93% (Figura 16). No problema *W* o agrupamento das placas e o escalonamento dos grupos pela heurística construtiva traz um ganho aproximado de 30 horas.

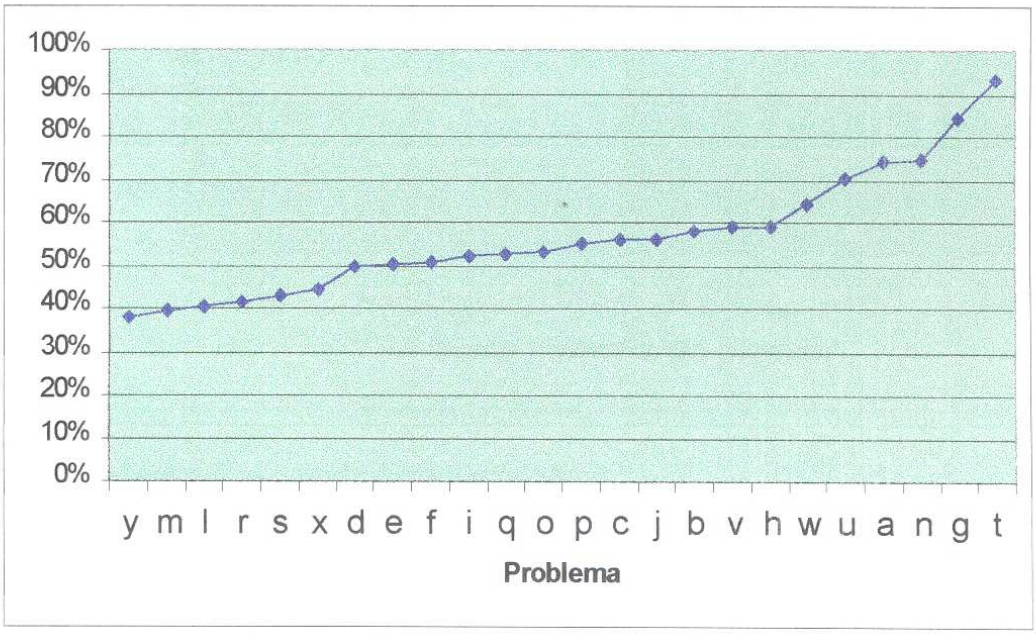


Figura 16. Ganho % no makespan obtido pela aplicação do algoritmo de formação de grupos e da heurística construtiva de escalonamento de grupos

O tempo computacional para o cálculo de uma solução inicial pela rotina Heurística é muito baixo. Pode-se dizer que a resposta é quase instantânea.

[8] Heurística para encontrar uma boa solução inicial descrita no capítulo 5.

6.4 Resultados das Rotinas Auxiliares

6.4.1 Tempo computacional da rotina Monta

Para avaliar o tempo computacional da rotina Monta foram feitos testes com as placas da seleção IV (Tabela 18). Cada placa de cada problema *(t, u, v, w, x, y)* foi considerada com um grupo com uma única placa, ou seja, não foi feito agrupamento. Foram medidos os tempos para 1000 chamadas da rotina Monta. Os tempos unitários aproximados para execução da rotina monta foi de 0,2 s (Figura 17).

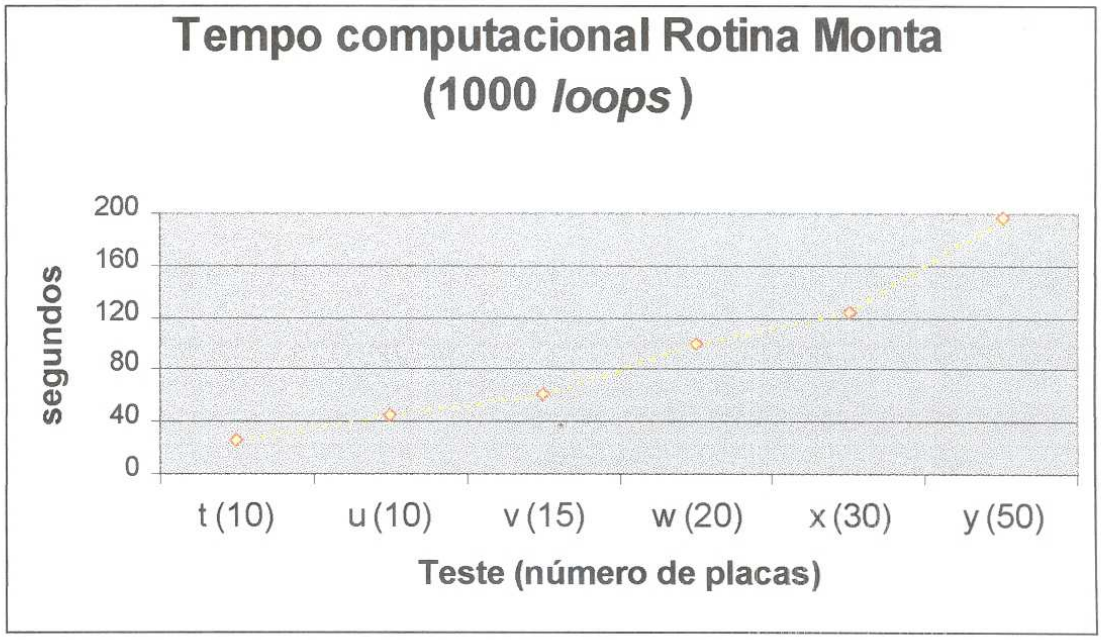


Figura 17. Tempo Computacional da Rotina Monta

6.4.2 Testes para determinar a temperatura inicial SA

Para calculara a temperatura inicial c_0, utilizada no SA, (item **5.5.2**) foi utilizada uma formula onde aparece a variável *percentual de aceitação*. Para determinar o valor desta variável foram feitos testes com percentuais variando de 30% a 90% (Tabela 21). Os testes foram realizados com as três estruturas de vizinhança descritas no item **5.5.2**.

Estes testes foram feitos com as placas da seleção IV (Tabela 18) problemas u (10 placas) e x (30 placas). Cada placa foi considerada como um grupo com apenas uma placa. Os resultados da Tabela 21 são referentes ao problema x. Com problema *y* os resultados foram semelhantes.

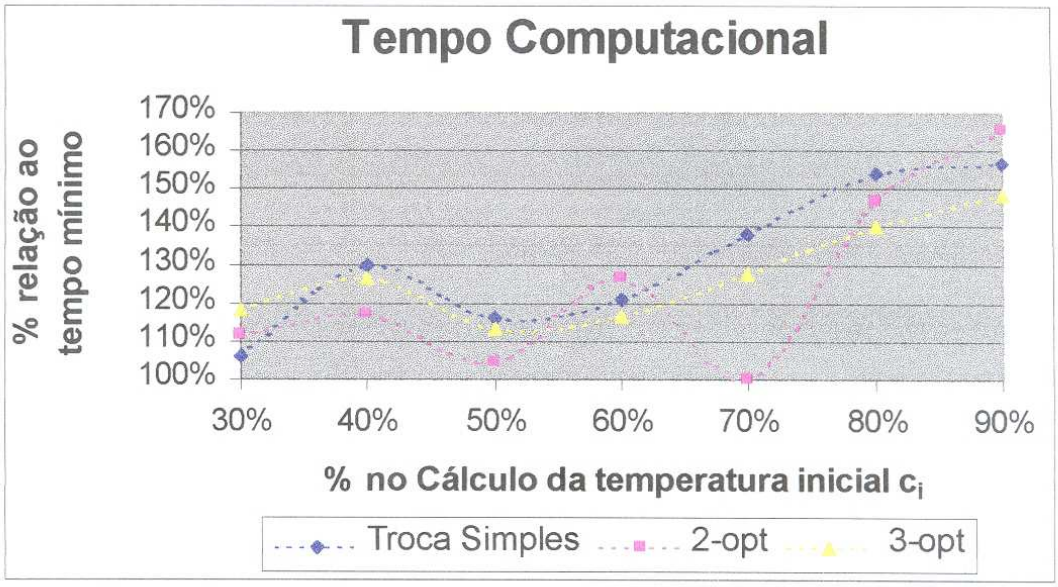
Os dados da Tabela 21 foram plotados em dois gráficos (Figura 18). Um mostra o tempo computacional e o outro mostra o *makespan* obtido a cada percentual. Ambos gráficos foram referenciados ao mínimo do tempo computacional e ao mínimo do *makespan.* O percentual de aceitação adotado foi de 70% por ter a melhor relação tempo computacional *versus makespan.*

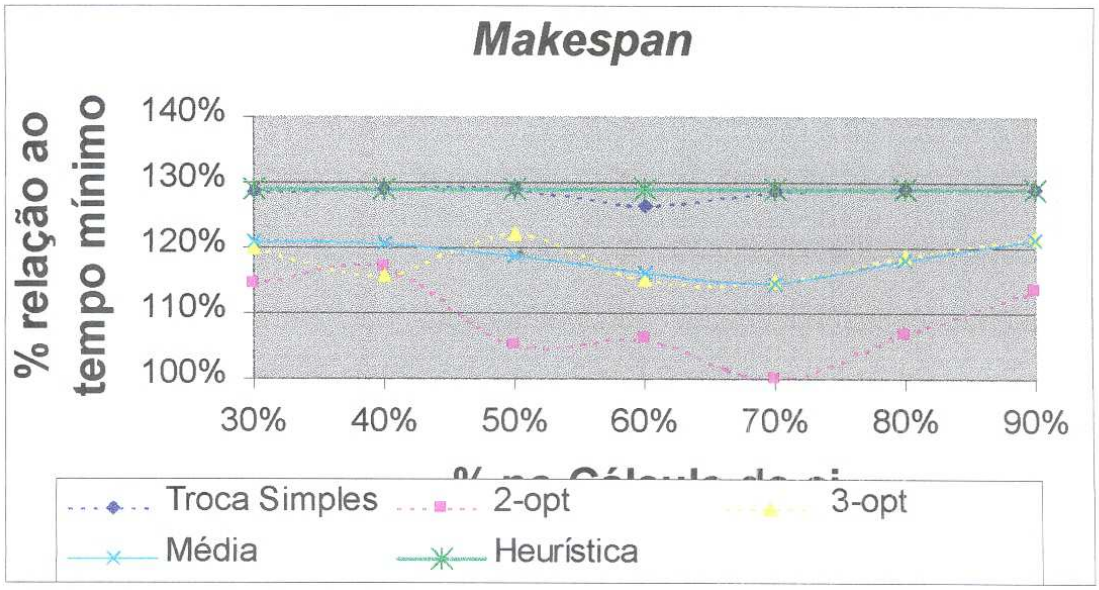
A estrutura de vizinhança que apresentou os melhores resultados foi a 2 – opt (Figura 18).

Tabela 21. Testes para determinação do % de aceitação

%	tempo			makespan			saída		
	troca	2-opt	3-opt	troca	2-opt	3-opt	troca	2-opt	3-opt
30%	0:32:15	0:33:51	0:35:50	77:02:29	68:36:25	71:56:26	cf	cf	cf
40%	0:39:20	0:35:35	0:38:24	77:11:23	70:05:19	69:26:04	cf	cf	cf
50%	0:35:09	0:31:41	0:34:26	77:11:23	63:00:32	73:13:19	cf	cf	cf
60%	0:36:39	0:38:27	0:35:28	75:48:53	63:31:10	69:02:54	cf	cf	350
70%	0:41:56	0:30:22	0:38:49	77:07:32	59:53:29	68:50:10	cf	cf	cf
80%	0:46:49	0:44:31	0:42:37	77:11:23	63:59:20	71:08:25	350	cf	cf
90%	0:47:37	0:50:16	0:45:01	77:11:23	67:58:23	72:44:49	cf	cf	350
Mínimo		0:30:22			59:53:29				
Heurística (vH)					77:11:23				

Tempo = tempo computacional;
Makespan = tempo total de produção de trinta grupos;
Saída = indica como o algoritmo SA parou, cf temperatura final, 350 = número de temperaturas sem trocas;





* média = (troca + 2-opt +3-opt)/3

Figura 18. Gráficos Tempo Computacional e *Makespan*

6.5 Testes com a algoritmos de programação da produção

Os testes realizados nos algoritmos de programação da produção tem como objetivo avaliar o desempenho do algoritmos sob dois pontos de vista: tempo computacional e melhoria do tempo total de produção.

Todos os problemas foram submetidos ao algoritmo SA, com as três estruturas de vizinhança mencionadas(Tabela 22). Verifica-se que os tempos computacionais praticamente não variam em função da estrutura de vizinhança (Figura 19). Com SA 2-opt o maior tempo computacional foi inferior a 19 minutos em todos os testes.

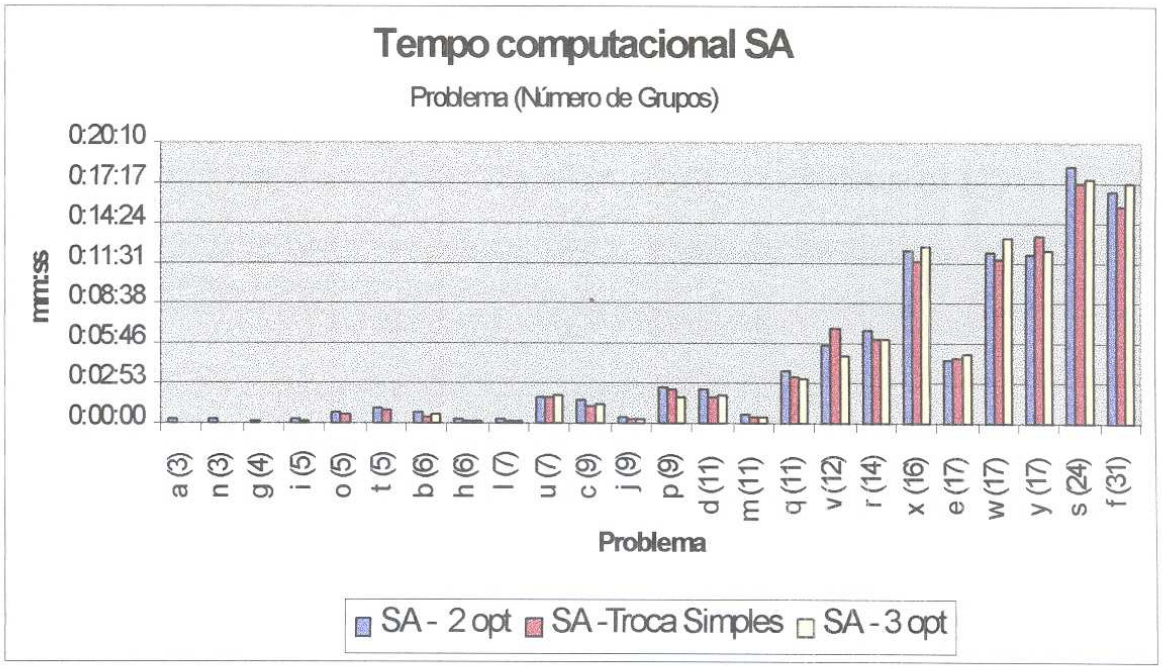


Figura 19. Tempo Computacional e Estruturas de Vizinhança SA

Com o problema *x* da seleção IV (Tabela 18), foram formados 30 grupos, cada grupo com apenas uma placa. . Os grupos foram submetidos ao algoritmo SA e foi anotado o número de trocas a cada temperatura e a redução do tempo total de produção. A estrutura de vizinhança usada foi 2-opt. A temperatura inicial foi $c_0 = (\Delta\ 10\ makespan)\ /\ \mathrm{Ln}(1\ /\ 0{,}7\ ))$. A solução inicial foi determinada

pela rotina heurística. Como pode ser verificado na Figura 20, o número de trocas a cada temperatura decai durante o resfriamento.

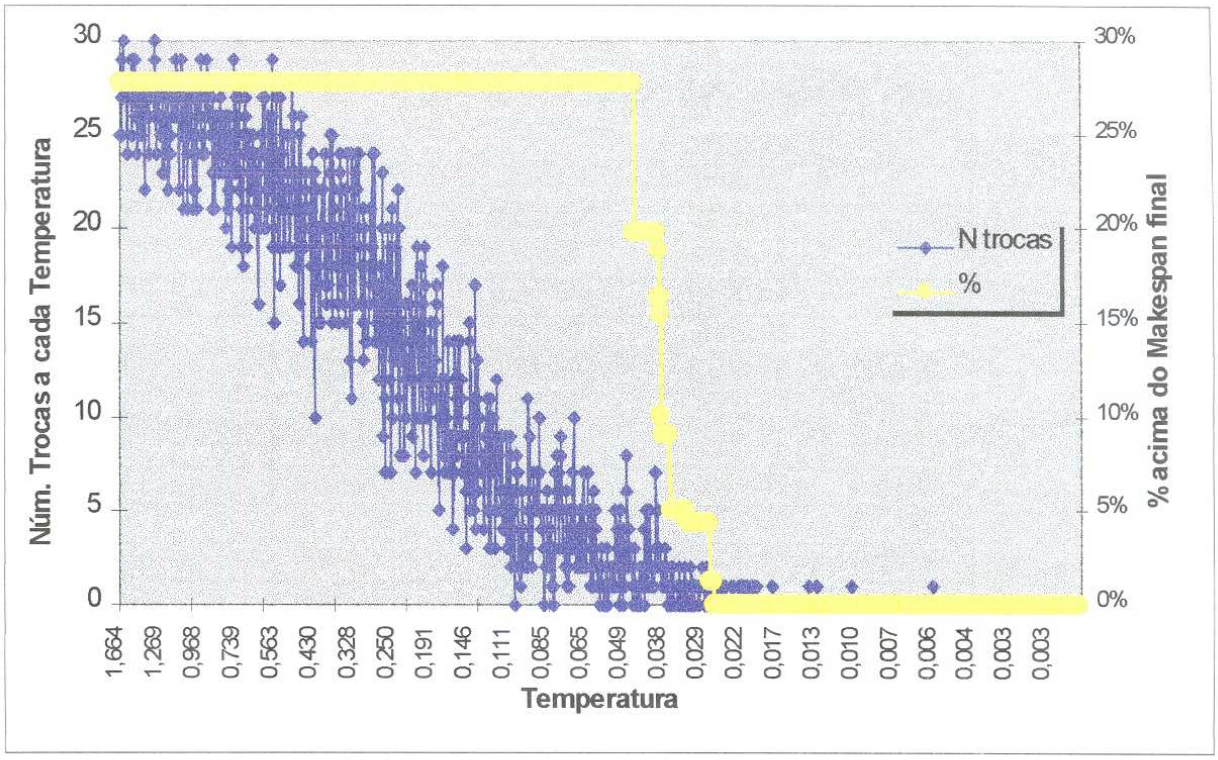


Figura 20. Número de trocas a cada temperatura e ganho % do algoritmo SA em relação a heurística construtiva aplicada ao problema *x*.

Conforme a Tabela 22 o algoritmo BB para os problemas *c, j* e *p* (todos com com nove grupos), apresentou tempos que variaram de 5 minutos a 20 horas. Esta variação nos tempos é função do número de ramos pesquisados. Foram feitas varias tentativas para melhorar a qualidade do *lower bound* (LB) a fim de diminuir o número de ramos pesquisados. Entretanto, estas alternativas foram infrutíferas. Assim, o uso do algoritmo BB deve ser feito com reservas quando o número de grupos for superior a seis.

A Tabela 24 mostra os tempos totais de produção (*makespan*) encontrados pelos algoritmos BB e SA com as três estruturas de vizinhança. Conforme já

comentado o SA com a estrutura de vizinhança 2-opt apresentou melhores resultados que a simples troca e o 3-opt.

Tabela 22. Tempos computacionais (*Sheduling*)

Problema (Núm. Grupos)	*BB*	*SA -Troca Simples*	*SA - 2 opt*	*SA - 3 opt*
a (3)	0:00:02	-	0:00:19	-
n (3)	0:00:04	-	0:00:22	-
g (4)	0:00:03	0:00:03	0:00:11	-
i (5)	0:00:07	0:00:06	0:00:14	-
o (5)	0:01:52	0:00:40	0:00:49	-
t (5)	0:02:32	0:00:53	0:01:03	-
b (6)	0:05:59	0:00:30	0:00:48	0:00:32
h (6)	0:00:32	0:00:06	0:00:18	0:00:08
l (7)	0:00:26	0:00:12	0:00:15	0:00:13
u (7)	1:45:20	0:01:47	0:01:51	0:01:55
c (9)	19:33:26	0:01:14	0:01:38	0:01:23
j (9)	0:05:12	0:00:16	0:00:27	0:00:14
p (9)	2:27:58	0:02:26	0:02:38	0:01:53
d (11)		0:01:51	0:02:24	0:02:02
m (11)		0:00:25	0:00:38	0:00:26
q (11)		0:03:19	0:03:51	0:03:14
v (12)		0:06:46	0:05:37	0:04:50
r (14)		0:06:01	0:06:39	0:06:04
x (16)		0:11:40	0:12:23	0:12:45
e (17)		0:04:41	0:04:37	0:05:04
w (17)		0:11:53	0:12:21	0:13:18
y (17)		0:13:27	0:12:06	0:12:29
s (24)		0:17:13	0:18:32	0:17:37
f (31)		0:15:33	0:16:38	0:17:21

Nas duas últimas colunas da Tabela 23 estão os ganhos percentuais em relação aos valores de *makespan* encontrados pela rotina Heurística. O ganho máximo foi de 37% no problema *l* (Figura 21).

Os ganhos globais encontrados comparando-se os tempos de produção das placas, sem a formação de grupos, (média de trinta soluções aleatórias, (Tabela 20)) e os tempos de produção após a formação dos grupos e o escalonamento da produção feito pelos algoritmos de BB ou SA (Tabela 23) chegaram a 238% (Tabela 24), como pode ser visualizado na Figura 22. O

tempo computacional total para o pior caso ficou inferior a 0,5 hora (6 minutos para formação de grupos mais 20 minutos para *scheduling*).

Tabela 23. Tempo Total de Produção

Problema	*Placas*	*Grupos*	*Heurística Grupos*	*BB*	*SA – Troca Simples*	*SA - 2opt*	*SA - 3opt*	*% Heurística. x BB*	*% Heurística x SA 2opt*	*Diferença entre BB e SA – 2opt*
a	5	3	13:43	12:16	12:16	-	-	12%	-	-
n	5	3	20:45	19:19	19:19	-	-	7%	-	-
g	5	4	5:30	5:15	5:15	5:15	-	5%	5%	0%
i	15	5	7:50	6:59	6:59	6:59		12%	12%	0%
o	10	5	33:20	31:03	31:03	31:03	-	7%	7%	0%
t	5	5	29:44	29:44	29:44	29:44	-	0%	0%	0%
b	10	6	19:46	18:41	18:41	18:41	18:41	6%	6%	0%
h	10	6	6:33	6:07	6:07	6:07	6:07	7%	7%	0%
l	30	7	11:18	8:16	8:16	8:16	8:16	37%	37%	0%
u	10	7	42:06	38:41	39:02	38:41	38:41	9%	9%	0%
c	15	9	28:00	25:44	26:34	26:28	26:28	9%	6%	3%
j	20	9	10:38	9:04	9:04	9:04	9:04	17%	17%	0%
p	15	9	47:52	40:27	41:58	41:27	43:20	18%	15%	2%
d	20	11	33:48	-	31:37	30:41	31:54	-	10%	-
m	50	11	18:01	-	15:42	15:42	15:42	-	15%	-
q	20	11	61:55	-	60:02	57:21	58:48	-	8%	-
v	15	12	51:03	-	46:46	46:57	46:12	-	10%	-
r	30	14	73:22	-	73:22	70:56	72:55	-	3%	-
x	30	16	51:31	-	50:30	48:03	49:12	-	7%	-
e	30	17	51:32	-	47:09	45:32	45:04	-	14%	-
w	20	17	53:24	-	51:39	50:37	49:35	-	8%	-
y	50	17	58:08	-	55:32	52:22	53:55	-	11%	-
s	50	24	125:36	-	125:36	122:08	125:36	-	3%	-
f	50	31	84:02	-	84:02	82:52	83:53	-	1%	-

Os tempos em negrito encontrados por SA foram iguais aos ótimos.

Tabela 24. Ganhos %: Solução aleatória X Solução obtida pelo sistema

Problema (no. Placas, no. Grupos)	Ganho %	Problema (no. Placas, no. Grupos)	Ganho %	Problema (no. Placas, no. Grupos)	Ganho %	Problema (no. Placas, no. Grupos)	Ganho %
a(5,3)	51%	g(5,4)	24%	n(5,3)	44%	t(5,5)	**7%**
b(10,6)	83%	h(10,6)	81%	o(10,5)	102%	u(10,7)	55%
c(15,9)	94%	i(15,5)	116%	p(15,9)	114%	v(15,12)	87%
d(20,11)	121%	j(20,9)	109%	q(20,11)	105%	w(20,17)	68%
e(30,17)	128%	**l(30,7)**	**238%**	r(30,14)	148%	x(30,16)	143%
f(50,31)	100%	m(50,11)	191%	s(50,24)	139%	y(50,17)	191%

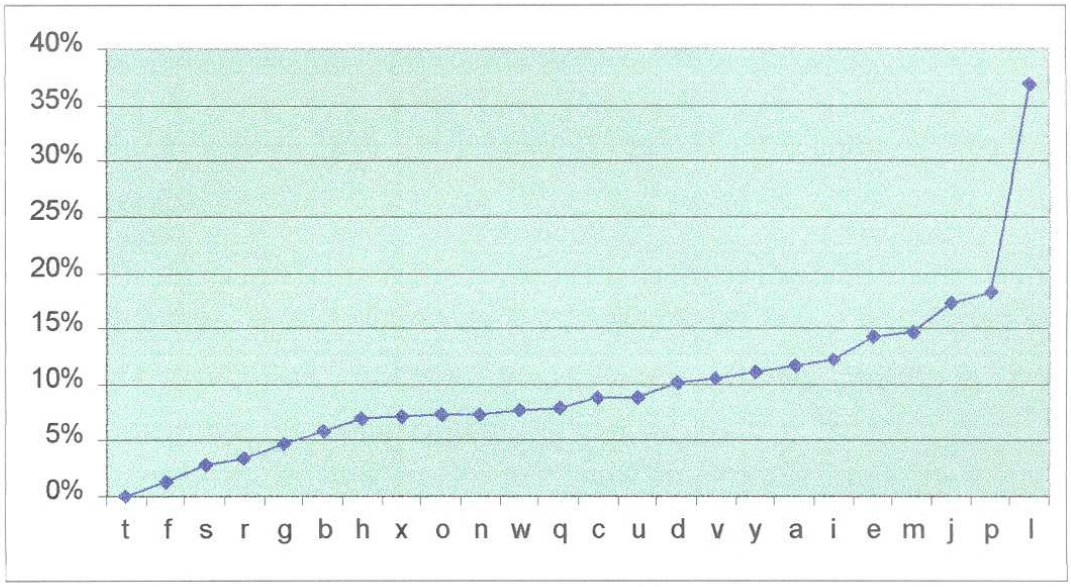


Figura 21. Heurística x (BB e SA)

Ganho % no *makespan* obtido pela aplicação do algoritmos de formação de grupos BB e heurística SA de escalonamento de grupos em relação aos valores encontrados pela heurística construtiva sem a formação de grupos

Figura 22. Ganho % no makespan

Ganho % obtido pela aplicação do algoritmos de formação de grupos, BB e heurística SA de escalonamento de grupos comparando-se com os resultados obtidos sem aplicação de algoritmo algum

6.6 Conclusões e pesquisas futuras

Este trabalho iniciou-se com um problema real de uma linha de produção de placas eletrônica.

Após vários testes os algoritmos de formação de grupos e escalonamento da produção mostraram-se robustos e confiáveis. Como facilidade adicional podem ser utilizados como ferramenta de simulação.

Futuras pesquisas poderiam ser realizadas no sentido de ampliar o espectro de cobertura dos modelos desenvolvidos, a fim de torná-los aplicáveis a uma variedade maior de ambientes produtivos de montagem SMT. Assim, um problema mais genérico pode ser assim enunciado:

Seja um problema com K de diferentes linhas de produção SMT ($k<=10$), p_k máquinas em cada linha K ($p_k <= 5$, $k= 1,2,...,k$) , f mesas de alimentadores ($f >= 2 \Sigma p_k$), b operadores de máquinas exclusivos, b' preparadores de mesas, e b" operadores capazes de operar máquinas e configurar mesas de alimentadores, n placas com vários componentes diferentes por placa, onde se conhece:

- o tamanho de lote de cada placa;
- a data de entrega de cada lote;
- a configuração de cada lote;
- a capacidade de cada mesa de alimentadores;
- a largura de cada alimentador;
- a largura de cada fita de componente e o espaço ocupado por cada carretel de componente no alimentador;
- os tempos de montagem e desmontagem dos alimentadores;
- as curvas de velocidade de cada cabeçote de montagem (tempo médio de montagem de cada placa).

Alem disso é assumido que:

- -preempção não é permitido;
- as mesas de componentes são similares mas não necessariamente idênticas. Estas mesas podem ser ou não ser intercambiaveis entre as máquinas;
- alguns componentes devem ser montados em máquinas de baixa velocidade e alta precisão;
- os componentes podem ser fornecidos em carreteis (tendência), ou a granel. Os componentes fornecidos a granel são montados em alimentadores especiais;

O objetivo é determinar ao escalonamento da produção (e a respectiva configuração das mesas alimentadores) de forma a minimizar a função multi-objetivo F tal que:

F= (makespan, atrazos, etc)

Referências Bibliográficas

BIXBY, Bob; APPLEGATE, David,; Rice University and AT&T Bell Labs; http://softlib.rice.edu/CRPC/newsletters/jan93/news.tsp.html, 1993

AARTS, E. H. L.; LENSTRA, J. K.. Local search algorithms. John Wiley & Sons, 1998

ARNOW, Dennis J. Increasing trhoughput a surface mount assembly line by improving machine balance and revising metrics. MIT – Massachusetts Institute of Technology, 1993

ASKIN, Ronald G.; STANDRIGE, Charles R.; Modeling and Analisys of Manufacturing Systems; John Willey, Sons; 1993

BLAZEWICZ, J.;Ecker, K.H.; SCHIMIDT, G.; Weglarz. Scheduling in computer and manufacturing systems. Second, Revised Edition, Springer Verlag, 1994

BRADLEY, Setphen P.;HAX, C. Arnoldo; MAGNANTI, Thomas L.. Applied Mathematical Programming. Addison-Wesley Publishing Company; 1977

BROCHONSKI, Paulo Cesar Brochonski, Candido, Marcos A. B.; Sistema Para Programação da Produção com Capacidade Finita em Máquinas SMT; SOBRAPO, 1998

CÂNDIDO, Marco Antonio Barbosa. A Hybrid Genetic approach to solve real make-to-order job shop scheduling problems. Doctoral Thesis; Universidade Federal de Santa Catarina, University Of South Florida; 1997

CANDIDO, Marco A. B.; Khator, Suresh K.; Bortolozzi, Flávio; Neto, Anselmo C.; Applying Simulated Annealing to Real Job Shop Scheduling Problems; Advances in Industrial Engineering Applications and Practice II, 1211-1216, 1998

CANDIDO, Marco Antonio Barbosa; Notas de aula do curso de Modelagem de sistemas; PUC – PR; 1998a

CANDIDO, Marco Antonio Barbosa; KHATOR S. K.; BARCIA R. M.; A genetic algorithm based procedure for more realistic job shop scheduling problems; INT. J. PROD. Res,. nº 12, 3437-3457; 1998

CANDIDO, Marco Antonio Barbosa; Brochonski, Paulo Cesar; An Intelligent Scheduling System for SMT Assembly Lines; FAIM99, Holand, 1999

CASTI, John L.. Five Golden Rules Grat Theories of 20th-Century Mathematics and Why They Matter;1996

COOK, Willian J., CUNNINGHAN, Willian J., SCHRIJVER, ALEXANDER, Willian R.. Combinatorial Optimization, John Wiley & Sons, 1998

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, L. Ronald; Introduction to Algorithms; MIT Press Cambridge, McGraw-Hill Book Company; 1990

CORNEL UNIVERSITY. The traveling salesman problem; Engineering Applications of OR, Handout. www.orie.cornell.edu/~or15/handouts/tsp/tsp.html; Fall 1996

CRAMA, Y; Kolen, A.W.J.; OERLEMANS A. G.; SPIEKSMA, Frits C. R.. Thoughput rate optimization in the automated assembly of printed circuit boards; Annals of Operations Research, 1990

CRAMA, Yves; FLIPPO, E. Olaf; KUNDERT, Joris van de; SPIEKSMA, Frits C. R.; A case with multiple machines and multiple board types. European Journal of Operational Research, 457-472; (1997) 1998

FÖHRENBACH, A.; GRUNOW, M.;Günther H. O.;SHORLING Ch.. Aggregation of product data for solving the daily order mix problem in small-lot PCB assembly. Uni. of Berlin. 22º Ind. Conf. on Computers and Industrial Engenering, Cairo, DEC/97

GARAY, M. R.; JOHNSON, D. S., Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, San Francisco, 1979

GOLDBERG, David E; Genetic algorithms in search, optimization, and machine learning. Addison Wesley; 1989

GRONALT, M.; GRUNOW M.; GÜNTHER, H.O.; ZELLER, R.; A heuristic for component switching on SMT placements machines. International Journal of Production Economics, Elsevier;JUL/97

GÜNTHER, Hans-Otto; GRUNOW, Martin; Schorling, Christopher; Uni. of Berlin; Workload planning in small lot printed circuit board assembly; OR Spektrum; Springer-Verlag, 1997

HOROWICZ, Ellis; SAHNI Sartaj; RAJEASEKARAN Sanguthevar. Computer algoritms. Computer Science Press; 1998

http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html

JOHNSON, David S., AT& T Labs.-Research, Florham Park; MCGEOCH, Lyle A., The traveling salesman problem: a case study; Local Search Optimization, Local Search in Combinatorial Optimization, Edited by E. Aarts and J.K. Lenstra, John Wiley & Sons; 1997

KARP, R. M.. Combinatorics, complexity and randomness, Comm. Of the ACM, v. 29, pp. 98-118, 1986

KIRKPATRICK, S., GELATT, C.D., VECCHI, M.P.; Optimization by Simulated Annealing. Science vol. 220, 671-680; 1983

LAARHOVEN, Peter J. M. Van; AARTS, Emile H. L.; LENSTRA, Jan Karel. Job Shop Scheduling by Simulated Annealing. Operations Reserch Society of America, FEV/1992

LEON, Jorge V.; BRETT, A. Petters. Replanning and analysis of partial setup stratégies in printed circuit board assembly systems. Department of industrial Engineering; Texas A&M Univerity; College Station, TX 77843-3131

MARTINICH, Joseph Stanislaus; Production and operations management: na applied modern approach;John Willey & Sons, 1997

MARTINS, Petrônio Garcia, Laugeni, Fernando P.; Administação da Produção; Editora Saraiva; 1998

MATTELD, Dirk C.; Evolutionary search and the job shop: investigation on genetic algorims for production scheduling; Heidelberg Physica Verlag; 1996

MCFREDIES, Paul; Excel 5 Super Book; Sams Publishing; Berkeley; 1994

MOREIRA, Daniel A.. Adiminstração da produção e operações. Editora Pioneira Adminstração e Negocios; 1998

PRESS, Wilian, FLENNERRY P. Brian; TEUKOLSKY, Saul A., VETTERLING, William T.;. Numerical Recipes in Pascal. University of Cambridge, 1990

REINELT, Gerhard; TSPLIB A traveling salesman problem library, ORSA, Journal on Computing 3; 376 – 384, 1991

SHTUB, Avraham; MAIMON, Oded. Grouping methods for printing cirtuit board assembly. INT. J. Prod. Res., 1379-1390 vol.:29; 1991

SHTUB, Avraham; MAIMON, Oded; Role of similarity in PCB grouping procedures; INT. J. Prod. Res., 973-983 nº 5; 1992

SOKAL, R. R., and SNEATH, P. H. A.. Principles of Numerical Taxionomy, A series de books in biology. San Francisco California: Freeman and Company; 1963

SULE, Dileep R.; Industrial Scheduling; PWS Publishing Company; Lybrary of Congress; 1996

TERRADA, Routo. Desenvolvimento de algoritmos e estruturas de dados. McGraw Hill, Makron; 1991

WAGNER, M. Harvey. Pesquisa operacional. 2º edição; Prentice-Hall do Brasil.

Glossário

- **Benchmark,** Termo que indica os "melhores resultados do mundo" dentre as organizações concorrentes, em determinados itens de controle;
- **EDD (Earliest Due Date),** Seleciona tarefa com a data de entrega mais próxima;
- **FIFO (First-in, first-out),** processa a tarefa que está a mais tempo na fila;
- **fine-pitch** – Componentes com pequeno espaçamento entre os terminais (menor que ~0,02mm);
- **LWKR (Least Work Remaining),** seleciona entre as tarefas que estão sendo processadas a tarefa com o menor tempo restante de processamento;
- **Makespan,** tempo total necessário para a montagem de uma seqüência de lotes de placas. Compreende desde o tempo de preparação necessário para o primeiro lote até a montagem do último elemento do último lote;
- **NP,** Nondeterministic Polinomial Time, é uma classe de problemas de desicão com a característica de serem não deterministas em um tempo polinomial.
- **PCI,** Placa de Circuito Impresso, é a placa onde são montados os componentes SMD.
- ***setup off line,*** quando a preparação da máquina pode ser feito em separado da máquina, permitindo assim que seja feita a preparação do próximo lote a ser montado enquanto a máquina esta montando o lote atual;
- ***Setup,*** tempo necessário para preparação de uma máquina para a produção;
- **SMD,** Sufarce Montage Device;
- **SMT,** Surface Montage Tecnology;
- **SPT, Shortest Processing Time,** seleciona a tarefa com o menor tempo de processamento;

Índice Remissivo