Cesar Scarpini Rabak

OTIMIZAÇÃO DO PROCESSO DE INSERÇÃO AUTOMÁTICA DE COMPONENTES ELETRÔNICOS EMPREGANDO A TÉCNICA DE TIMES ASSÍNCRONOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia

São Paulo
1999

Rabak, Cesar Scarpini

Otimização do Processo de Inserção Automática de Componentes Eletrônicos Empregando a Técnica de Times Assíncronos. São Paulo, 1999.
114 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Elétrica.

1. Times Assíncronos – A-Teams 2. Otimização Discreta 3. Inserção Automática de Componentes Eletrônicos. 4. Problema do Caixeiro Viajante – TSP 4. Problema da Atribuição Quadrática - QAP I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Elétrica. II. t

Cesar Scarpini Rabak

OTIMIZAÇÃO DO PROCESSO DE INSERÇÃO AUTOMÁTICA DE COMPONENTES ELETRÔNICOS EMPREGANDO A TÉCNICA DE TIMES ASSÍNCRONOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia

Área de Concentração:
Sistemas Digitais

Orientador:
Prof. Dr. Jaime Simão Sichman

São Paulo
1999

AGRADECIMENTOS

Ao orientador Prof. Dr. Jaime Simão Sichman pelas diretrizes seguras, e pelo apoio e incentivo.

À Célia, minha esposa, por todo o estímulo e interminável compreensão.

Ao Instituto de Pesquisas Tecnológicas de São Paulo pelos recursos disponibilizados.

Ao Laboratório de Técnicas Inteligentes pela acolhida e recursos materiais colocados à minha disposição.

Aos amigos Casimiro, Márcio, Nilsa, Orlando, Pedro e Rigo pela amizade e enriquecedoras discussões.

Lista de Figuras

Lista de Tabelas

Sumário

RESUMO

Máquinas insersoras de componentes são utilizadas na indústria eletrônica moderna para a montagem automática de placas de circuito impresso. Com a competição acirrada, há necessidade de se buscar todas as oportunidades para diminuir custos e aumentar a produtividade na exploração desses equipamentos.

Neste trabalho, foi proposto um procedimento de otimização do processo de inserção da máquina insersora AVK da Panasonic, implementado em um sistema baseado na técnica de times assíncronos (A-Teams).

Foram realizados testes com exemplos de placas de circuito impresso empregadas por uma indústria do ramo e problemas sintéticos para avaliar o desempenho do sistema.

ABSTRACT

Component inserting machines are employed in the modern electronics industry for the automatic assembly of printed circuit boards. Due the fierce competition, there is a need to search for all opportunities to reduce costs and increase the productivity in the exploitation of these equipment.

In this work we propose an optimization procedure for the insertion process of the AVK Panasonic inserting machine, implemented in a system based on asynchronous teams (A-Teams).

Tests were conducted using as examples both printed circuit boards used by a particular industry of the realm and synthetic problems for the evaluation of the system.

1 INTRODUÇÃO

Neste capítulo, inicialmente situamos o leitor no contexto deste trabalho, apresentamos seus objetivos, motivação e resultados esperados e concluímos com a descrição da organização do restante do documento.

1.1 Contexto do Trabalho

Máquinas insersoras de componentes são utilizadas na indústria eletrônica moderna para a montagem automática de placas de circuito impresso. Essas placas formam a parte principal dos equipamentos eletrônicos tanto de entretenimento (tais como rádios, televisores, vídeo cassetes), como de informática (tais como microcomputadores, monitores de vídeo, máquinas de FAX, etc.). Com a competição acirrada nesses mercados, há necessidade de se buscar todas as oportunidades para diminuir custos e aumentar a produtividade na produção desses bens.

A fase de montagem automatizada é uma área onde existem oportunidades para se obter potencialmente grandes economias. No Brasil, em particular, as máquinas de montagem automática (das quais as máquinas insersoras são um caso particular) têm preço relativamente alto, o que torna o seu custo de operação uma componente importante na matriz de custos de produção dos equipamentos eletrônicos.

A máquina insersora, também denominada na literatura de "pick-and-place", consiste tipicamente de um fixador para a placa de circuito impresso, um conjunto de alimentadores (escaninhos) e uma cabeça de inserção, cuja

função é retirar componentes dos alimentadores e inserir na placa. A unidade de controle da insersora recebe uma série de instruções codificadas, ao qual se denomina de programa de inserção, e executa essas instruções[1]. O programa de inserção indica para cada instrução a posição do componente na placa, o tipo de componente (ou seja, de qual escaninho deve ser pinçado), e operações adicionais que possam ser necessárias. O tempo de ciclo da máquina é o tempo necessário para executar completamente uma instrução. O tempo total de inserção de cada placa é a soma dos tempos de cada instrução mais um tempo (geralmente fixo) de alimentação de cada nova placa (vazia) na máquina. Esse tempo total é influenciado pela ordem da inserção dos componentes na placa e pela preparação dos escaninhos, ou seja, pela forma como os diversos componentes são designados aos escaninhos.

O objetivo da otimização é minimizar o tempo total de inserção, aumentando assim a utilização da máquina insersora. A otimização é obtida escolhendo-se a ordem apropriada de inserção dos componentes na placa, o que corresponde a resolver uma instância do *problema do caixeiro viajante*[2] [LAW85] ("Traveling Salesman Problem"), e a colocação mais apropriada dos componentes nos escaninhos, que pode ser enunciado como o *problema de alocação quadrática*[3] [LAW63] ("Quadratic Assignment Problem"). Tanto um como outro problema são conhecidos pela dificuldade de solução devido à explosão fatorial das soluções, sendo caracterizados como problemas do tipo $\mathcal{NP}$-difícil [GAR79].

Um dos grandes problemas que se enfrenta nesses problemas de otimi-

[1]Na literatura especializada nesse tipo de máquina, as instruções são também chamadas de blocos do programa de inserção.

[2]Por concisão, doravante referiremos a este problema por TSP.

[3]Por concisão, doravante referiremos a este problema por QAP.

zação discreta é a ausência de algoritmos que sejam confiáveis e rápidos. Os mais rigorosos são lentos ou consomem muitos recursos computacionais, enquanto os heurísticos não são muito confiáveis, sendo sensíveis à instância do problema: dão bons resultados para alguns casos e falham fragorosamente em outros casos. Quando é possível identificar "a priori" a aplicabilidade desses algoritmos a um certo problema, essa sensibilidade acaba sendo resolvida. Entretanto, nem sempre essa identificação é exeqüível, sendo em muitos casos essa determinação um problema de solução tão difícil quanto a própria otimização buscada.

Ao invés de buscar novos e mais apropriados algoritmos, pode-se tentar combinar algoritmos já conhecidos para que eles cooperem na geração da resposta desejada. A nova questão passa a ser então como organizá-los para que eles interajam e produzam a solução buscada.

Uma nova abordagem para resolver essa questão é o uso de uma técnica denominada Times Assíncronos ("*Asynchronous Teams*")[4] [TAL96]. Nessa abordagem, cada algoritmo torna-se um agente e agentes são agrupados em fluxos de dados fortemente cíclicos. Os agentes podem desempenhar dois papéis diferentes: ou podem gerar novas soluções viáveis, ou podem retirar soluções menos viáveis ou que não atendam a algum critério. Aos primeiros dá-se o nome de construtores, e aos últimos dá-se o nome de destruidores. Com essa abordagem, tem-se um equilíbrio entre a produção e eliminação de hipóteses de solução.

Em um A-Team, os agentes são autônomos, e portanto não respondem a nenhum mecanismo de coordenação central. A cooperação entre eles é

[4]Por concisão, doravante passaremos a abreviar *Asynchronous Teams* como A-Teams.

assíncrona. Assim, caso haja recursos suficientes, como por exemplo vários computadores diferentes, os agentes podem rodar em paralelo. Esta técnica difere de outras abordagens para resolução de problemas que necessitam incluir restrições de precedência para forçar uma ordem (mesmo que parcial) nas atividades dos seus módulos computacionais (mesmo que sejam agentes). A experiência com essa técnica tem mostrado que algoritmos trabalhando dessa maneira são capazes de produzir em conjunto resultados muito melhores daqueles que seriam obtidos caso estivessem trabalhando sozinhos [KAN97] [TAL96].

1.2 Objetivos do Trabalho

Este trabalho tem como objetivo testar a eficácia da técnica dos A-Teams em um problema de otimização do processo de inserção automática de componentes eletrônicos em placas de circuito impresso.

Para tal, propomos uma abordagem baseada nesta técnica para resolver o problema de otimização de uma máquina insersora de componentes axiais Panasert AVK de fabricação pela empresa Panasonic Manufacturing Equipment Division[5] [PAN96], mostrada na figura 1. Essa abordagem permite o desenvolvimento de uma solução flexível, que aproveita a experiência e o trabalho já realizado por outros pesquisadores. Ela possibilita ainda que novos algoritmos que porventura venham a surgir possam ser acrescentados aos A-Teams, aumentando assim a precisão e abrangência das soluções produzidas.

[5]Por concisão, doravante utilizaremos apenas a designação AVK ao nos referirmos a esse equipamento.

O trabalho consiste na formulação matemática do problema, na implementação de um sistema computacional que o soluciona (sistema OPTIMA) e na validação experimental através da resolução de problemas de inserção gerados tanto sinteticamente como também usando-se casos reais de placas de produção efetivamente empregadas na montagem de equipamentos eletrônicos.

Figura 1: Máquina Insersora AVK

1.3 Motivação e Resultados Esperados

A otimização de qualquer processo industrial é em essência a "raison d´être" da Engenharia, sendo naturalmente um alvo em todos os tipos de indústrias. A rapidez das mudanças na indústria eletrônica impede que se despenda muito tempo na elaboração de soluções ótimas de forma manual, já que elementos como a dinâmica do JIT ("*Just in Time Manufacturing*"), ciclos de vida dos

produtos mais curtos, e exigência de taxas de produção mais altas e com lotes menores solicitam a obtenção dessas soluções em um espaço de tempo mais curto. Crama et al. [CRA90] afirmam: "...é hoje bem reconhecido que a disponibilidade de procedimentos automatizados de planejamento é uma grande vantagem." Um outro ponto importante, apresentado em Leipälä e Nevalainen [LEI89], observa que a otimização realizada por meios manuais (denominada de "otimizador humano") não é nada trivial, uma vez que geralmente envolve mais de um critério a ser otimizado, sendo difícil criar uma representação gráfica que permita ao operador visualizar um caminho mínimo.

Devido a essa importância econômica, um número muito grande de pesquisas destinadas à modelagem e otimização dos processos automáticos de montagem de circuitos impressos é encontrado na literatura. Dois autores chegam a citar que a automatização da montagem é a única solução para sustar a transferência de empregos de seus países para países estrangeiros [CHA87].

A solução de problemas de otimização de inserção de componentes em placas de circuito impresso é um problema que tem aplicabilidade imediata na indústria, cuja solução começa a ser abordada recentemente [BRO96] [MCG92]. No caso de máquinas como a AVK, onde essa otimização envolve a solução simultânea de um problema TSP e QAP e para o qual não há uma formulação ou abordagem geral, a técnica de A-Teams pode se mostrar adequada, considerando problemas de otimização em outros domínios que já foram solucionados com ela tais como: escalonamento de trens [KAN95], escalonamento de corridas de laminadores na indústria siderúrgica [IBM96b], escalonamento de máquinas de papel e otimização do corte de papel [IBM96a].

A própria solução do TSP puro foi utilizada por [SOU93] na sua tese de doutorado quando demonstrava a eficiência de escala (mais agentes tendem a melhores soluções) [KAN97].

Como resultados deste trabalho, esperamos obter uma formulação matemática genérica do problema de inserção de componentes eletrônicos em máquinas do tipo AVK, e a disponibilização de um sistema computacional baseado na técnica de A-Teams para resolvê-lo. Pretendemos ainda mostrar a eficácia da utilização desta técnica na resolução deste problema, através da análise de casos de teste e sua posterior comparação com resultados reais da indústria.

1.4 Organização do Documento

No capítulo 2 apresentamos uma visão geral do processo de montagem de placas de circuito impresso, com ênfase na inserção automática de componentes, foco desta dissertação, descrevendo os aspectos do emprego das máquinas automáticas utilizadas na montagem. Damos ainda uma classificação das insersoras que nos auxiliará a identificar o problema de otimização associado, relatado no capítulo 3. Descrevemos também a máquina insersora AVK, objeto de estudo deste trabalho, apresentando inclusive uma análise de seu funcionamento e de seus tempos de operação.

O capítulo 3 discute os aspectos do processo de otimização da inserção automática de componentes eletrônicos, analisando a decomposição desse problema em dois problemas de otimização discreta clássicos e avalia as dificuldades para a abordagem desses problemas. A formulação do problema

de inserção automática sendo estudado é então apresentada ao final deste capítulo.

No capítulo 4 descreve-se a técnica adotada neste trabalho para solucionar o problema proposto, mostrando-se os conceitos básicos dos A-Teams, bem como o protocolo de projeto de soluções baseadas nesta técnica. Mostramos ainda como essa técnica mostra-se adequada para abordar o problema específico de inserção automática de componentes que nos interessa.

O capítulo 5 descreve o sistema OPTIMA, baseado em A-Teams, cujo objetivo é a otimização da inserção de componentes eletrônicos da máquina insersora AVK. Indicamos a arquitetura escolhida, os algoritmos selecionados para empregar nos agentes autônomos, e o procedimento adotado para transformar estes algoritmos em agentes, tal como requerido pela técnica.

No capítulo 6 apresentamos os testes realizados com o sistema OPTIMA, e no capítulo 7 apresentamos nossas observações finais e sugestões para trabalhos futuros.

Como informação adicional, apresentam-se no apêndice A o formato do arquivo de programa de inserção da AVK, e no apêndice B o script utilizado para gerar as matrizes correspondentes ao QAP a partir de um arquivo de programa de inserção.

2 INSERÇÃO AUTOMÁTICA de COMPONENTES ELETRÔNICOS em PLACAS de CIRCUITO IMPRESSO

Neste capítulo, são apresentados os conceitos básicos ligados à montagem de circuitos impressos, descrevendo suas principais fases e dando atenção especial às operações de inserção automática de componentes, de modo a melhor contextualizar o trabalho proposto.

2.1 Processo de Montagem de Placas de Circuito Impresso

Placas de circuito impresso consistem de placas de suporte não condutor (geralmente plásticos como fenolite ou epóxi), com espessura entre 1,5 a 2,0 mm, contendo trilhas de material condutor (geralmente cobre na sua superfície). No caso de placas multicamadas, estas trilhas localizam-se tanto na superfície como em camadas "sanduichadas" [COO67] [SCA70]. Cada placa tem definida pelo seu projeto as posições onde componentes eletrônicos, tais como resistores, capacitores, diodos, transistores ou circuitos integrados, deverão ser montados para que ela execute a sua função no sistema a que será destinada.

Existem duas tecnologias distintas para fixar os componentes eletrônicos nas placas:

Through Hole

Os componentes eletrônicos têm seus terminais disponibilizados por meio de fios metálicos. As placas de circuito impresso contêm furos nos quais esses fios são introduzidos, tendo em volta destes ilhas de material condutor ao qual os fios são soldados, completando a montagem desse tipo de componente[6].

Surface Mount Technology (**SMT**)

Os componentes têm seus terminais formados de minúsculas plataformas de material condutor rente à sua parte inferior, denominadas *"pads"*. As placas de circuito impresso para montagem desta classe de componentes contêm ilhas correspondentes a cada terminal do componente, sem nenhum furo e a soldagem é feita a partir do topo, fixando os componentes à placa pela solidificação do filme de solda *entre* os *pads* do componente e as ilhas.

Existe um forte movimento da indústria de componentes de passar a usar essa segunda tecnologia, uma vez que a furação da placa na fase da fabricação da placa e a dobra e corte dos fios na fase de montagem podem ser prescindidas. Entretanto, certos componentes eletrônicos, devido às suas características físicas, ainda são unicamente disponibilizados na tecnologia *through hole*.

As fases que compõem a montagem dessas placas são tipicamente as seguintes [SHE86]:

Liberação de ordem de produção Determina-se o tamanho e data de ca-

[6]No caso de componentes muito grandes para ficarem estáveis apenas pela soldagem de seus terminais, pode haver também uso de parafusos ou rebites para prender a peça.

da ordem de produção conforme o escalonamento definido pelo sistema de planejamento de materiais, logística e eventualmente necessidades do mercado.

Kitting. Os componentes necessários a cada ordem de produção são designados e juntados em *kits*. As empresas geralmente empregam internamente sistemas de informação que identificam os componentes, peças e outros insumos de fabricação pelos chamados *part numbers*. Nesta fase, os componentes disponíveis em estoque e aqueles recebidos durante o planejamento JIT são separados tanto física como logicamente (eventualmente dando "baixa" nos registros de estoque).

Preparação Marcam-se as placas de circuito impresso para futuro rastreamento e operações preparatórias para a montagem, tais como secagem de placas em estufas, transferência de componentes de embalagem a granel para alimentadores específicos (para cada tipo de máquina ou equipamento), etc.

Inserção Efetiva-se a colocação dos componentes na placa de circuito impresso. A montagem dos componentes *through hole* requer a inserção de fios (ou pinos) desses componentes nos furos da placa. No caso dos componentes SMT, posicionam-se os terminais (*pads*) em correspondência com as ilhas de material condutor na placa. No caso dos componentes axiais, antes da inserção, é necessário proceder à dobra e ao corte desses fios. Os componentes SMT devem ser fixados com cola para prendê-los na placa de forma temporária.

Soldagem Realiza-se o processo de solda dos componentes colocados na placa. A conexão elétrica desses componentes é estabilizada pela soldagem dos fios ou terminais às ilhas por meio de solda branda feita de

uma liga eutéctica a base de chumbo e estanho [FIN75]. Esta operação tipicamente utiliza máquinas de solda automáticas do tipo onda de solda, onde a placa passa pela onda (tangenciando-a pela face oposta aos componentes no caso da tecnologia *through hole*) a uma determinada velocidade. Esta fase pode também incluir operações manuais, quando um operador, através do uso de um ferro de soldar, realiza a ligação elétrica entre a ilha da placa e o fio ou pino do componente utilizando solda formada em arame dispensada por um carretel, ou procedendo a imersão em um cadinho de solda fundida por meio de um bastidor basculante.

Teste Realizam-se testes para detectar defeitos na montagem. A placa é colocada em "jigas", equipamentos que a estimulam eletricamente de forma similar ao equipamento a que se destina. A aparelhagem (no caso de teste automático) ou o técnico (no caso manual) verifica se o comportamento da placa recém montada é aquele esperado. Outra metodologia, também denominada "*burn in*", destina-se a eliminar a mortalidade infantil dos componentes eletrônicos, quando se coloca a placa em condições similares às de operação em estufas nas quais a temperatura é ciclada nos limites de projeto. O objetivo é o de forçar os componentes com defeitos marginais a apresentarem defeitos mais rapidamente[7]. Finalmente verifica-se a funcionalidade da placa, onde se examina se ela atende às características de desempenho (tensões de saída, formas de onda, restrições de consumo, etc.) e se todos os circuitos nela contidos estão operando satisfatoriamente.

[7]Nos casos de placas destinadas a equipamento de maior responsabilidade (militar, aeronáutico, automotivo), essa fase de *burn in* pode incluir excitações mecânicas para simular vibração, choques, e eventualmente campos eletromagnéticos.

Devido à multitude e diferentes tecnologias com que são fabricados, os componentes eletrônicos adquirem uma grande variedade de formato, tamanho, embalagem. Por outro lado, componentes eletrônicos fisicamente idênticos quanto aos seus aspectos externos podem ser diferentes eletricamente (resistores de valor ôhmico diferente, ou transistores de mesmo encapsulamento mas com características elétricas diferentes, por exemplo). Essa variedade torna a montagem dessas placas uma operação complexa: certas tecnologias de componentes tornam a montagem manual de certos componentes praticamente impossível, enquanto outros tipos de componentes somente podem ser montados manualmente a custos toleráveis pela indústria no presente estado da tecnologia. São necessárias, em geral, duas etapas na montagem dessas placas: uma manual e outra automatizada. Dependendo do tipo de produto a que a placa de circuito impresso se destina, essa etapa automatizada pode chegar a 100% da fase de inserção.

2.2 Inserção Automática de Componentes Eletrônicos

A indústria eletrônica dispõe de máquinas automáticas que são capazes de efetuar a inserção de certos tipos (classes) de componentes, permitindo que esse processo de montagem seja mecanizado. Essas máquinas podem variar desde robôs de uso geral empregados para manusear os componentes eletrônicos (quando são providos de pinças e acessórios especiais), até máquinas especializadas, construídas especialmente para esse fim. Essas últimas são denominadas máquinas **insersoras**[8] de componentes eletrônicos.

[8]No caso de máquinas específicas para a tecnologia de componentes de montagem em superfície SMT, a palavra insersora deveria ser substituída por posicionadora. Contudo é usual no meio denominar-se essas máquinas também de insersoras.

Existem máquinas insersoras muito distintas que atendem tanto às necessidades da tecnologia específica dos componentes que inserem, como às diferentes abordagens de resolução de problemas tecnológicos que surgem na construção desse tipo de equipamentos.

Entretanto, pode-se observar que todas as máquinas insersoras executam cinco operações fundamentais:

[1] posicionamento para retirada de um componente de um alimentador;

[2] retirada do componente do alimentador corrente;

[3] transporte do componente desde o alimentador até a placa de circuito impresso;

[4] posicionamento para colocação do componente na placa de circuito impresso;

[5] inserção do componente na placa de circuito impresso.

O passo [3] pode envolver algum processamento intermediário, como por exemplo, rotação do componente, preparação dos fios (corte e/ou dobra), etc.

Em algumas máquinas, existe apenas um único alimentador de componentes. Neste caso, na fase de preparação os componentes são arranjados em uma única fita de alimentação, em uma ordem pré-definida correspondente à sua colocação pela insersora na placa de circuito impresso. Em outra classe de máquinas, os componentes a serem usados na montagem são disponibilizados em alimentadores individuais, um para cada tipo de componente. Neste outro caso, a fita de alimentação já faz parte da forma de condicionamento

do componente providenciada pelo fabricante. Esses alimentadores são chamados de **escaninhos** e podem ser montados em conjuntos dispostos tanto de forma linear como circular. Nestas máquinas, o posicionamento desses conjuntos para a retirada de um componente de um alimentador acarreta a movimentação de uma parte da máquina relativamente pesada, resultando em uma operação cara do ponto de vista de tempo. O número de escaninhos de uma máquina desse tipo torna-se uma de suas características, já que limitam o número máximo de tipos de componente que é possível colocar em uma placa em uma dada operação de inserção.

Ao classificar as máquinas insersoras, utilizamos o termo **ciclo de máquina** para designar a série de operações fundamentais necessárias para levar a cabo a retirada e inserção de um componente. As máquinas são diferenciadas pelos mecanismos utilizados para efetuar essas operações.

2.3 Classificação de Máquinas Insersoras

Dentre as características utilizadas para classificar as máquinas insersoras, a mais importante noção é a da **concorrência**, na qual se distinguem duas categorias:

1. **seqüenciais**: as cinco operações fundamentais descritas na seção 2.2 são executadas seqüencialmente, um ciclo consiste claramente dos passos fundamentais realizados naquela ordem;

2. **concorrentes**: dois ou mais operações fundamentais podem ser realizadas concorrentemente, um mesmo ciclo pode consistir de operações

fundamentais referentes à retirada de um componente e à inserção do componente previamente retirado.

Máquinas concorrentes podem ter ferramentas especializadas para retirada e posicionamento de componentes, com algum mecanismo de transporte entre os dois, ou possuir múltiplas cabeças de inserção, cada uma capaz de realizar as operações de retirada e inserção.

Mesmo as operações fundamentais podem ser mais ou menos concorrentes. O posicionamento para colocação do componente na placa de circuito impresso é tipicamente realizado por uma mesa XY que movimenta a placa, e os movimentos nos eixos também podem ser concorrentes ou seqüenciais.

A categoria de uma máquina afeta claramente o cálculo de seu tempo de ciclo da máquina. No caso de uma máquina totalmente seqüencial, este tempo será a soma das durações de cada operação fundamental. No caso de uma máquina concorrente, será necessário definir melhor o tempo de ciclo, verificando as operações concorrentes e os momentos em que ocorrem dependências entre elas. Por exemplo, se em uma determinada máquina o posicionamento da placa pode ser feita em paralelo com a retirada do componente do alimentador, o tempo de ciclo conterá um termo com o máximo entre esses dois tempos, uma vez que a inserção na placa requer que *ambas* as operações já tenham ocorrido; em outras palavras, é necessário estudar as precedências do ciclo dessas máquinas para formular os seus tempos de ciclo.

2.4 Níveis de Decisão de Planejamento

Uma vez que os aspectos técnicos da(s) tecnologia(s) envolvendo a montagem de placas tenham sido dominados, o **planejamento do processo** [MCG92] passa a ser a principal componente na determinação da produtividade na montagem de circuitos impressos. A importância dada a esta fase pode ser avaliada tanto pela literatura, como pela oferta das empresas (especialmente de informática) de soluções para planejamento de produção.

Um certo número de decisões resulta da interação entre o planejamento de compras e o planejamento e controle da produção (PCP). McGinnins et al. [MCG92] estabeleceram uma hierarquia dessas decisões separando-as em três níveis:

- Nível 1: **(Agrupamento)** Seleção dos grupos de máquinas e famílias (classes) de peças a esses grupos.
- Nível 2: **(Alocação)** Alocação dos componentes a máquinas quando um grupo tem mais de uma máquina.
- Nível 3: **(Arranjo e Seqüenciamento)** Arranjo dos alimentadores de componentes e seqüência das operações de colocação para cada máquina e circuito impresso.

Os autores dessa hierarquia enfatizam que essas decisões não são independentes, e mais importante, torna-se essencial que o processo seja automatizado dada a grande quantidade de informação a tratar.

As decisões de nível 1 incluem o escalonamento de máquinas adequadas

para cada tecnologia de componentes (SMT, radiais, axiais), principalmente quando uma certa máquina pode tratar de mais de um tipo de tecnologia.

O nível 2 engloba os casos onde máquinas têm capacidades diferentes, ou então onde se exceda a capacidade de alguma máquina, devido a alguma exigência de uma certa placa de circuito impresso. Como conseqüência, pode ser necessário particionar alguma operação em sub-fases. Por exemplo, como já citado anteriormente, o número de escaninhos de uma máquina limita o número de tipos de componentes distintos que é possível montar em uma única "passada" de inserção. Caso uma certa placa exija um número maior de tipos de componentes que a capacidade da máquina, será necessário particionar o problema em diversas passadas, tantas quantas forem necessárias para que essa parte da montagem seja terminada.

No nível 3, tem-se a programação das máquinas insersoras propriamente ditas (quando esta fase é automatizada) ou planejamento da linha de produção (quando as operações são manuais). Adotando uma visão bem macroscópica, esta fase pode ser vista como um problema de escalonamento envolvendo a seqüência na qual os componentes necessários são colocados na placa. Do ponto de vista tecnológico, fazer um programa de inserção que não viole as regras mecânicas (e de sintaxe) da máquina gera um programa correto. Contudo, as decisões deste nível envolvem preocupações com a otimização desse processo, ou seja, a elaboração de um programa de inserção correto que conduza a uma maior utilização da máquina, aumentando a produção e diminuindo o seu custo de operação.

Segundo Ball e Magazine, nesse nível encontram-se as maiores potencialidades para a redução de custo da produção [BAL88].

2.5 A Máquina Insersora Panasert AVK

A insersora Panasert AVK é uma máquina de fabricação da Panasonic [PAN96] para inserção de componentes *through hole* axiais, isto é, aqueles que têm seus fios alinhados com seu eixo longitudinal (tais como diodos, resistores, capacitores, supressores de transientes, etc.), em placas de circuito impresso que tenham furos para acomodar esses fios e receber solda pelo lado oposto a aquele por onde esses componentes tenham sido posicionados na placa. Conseqüentemente, para efetuar uma inserção de um componente a AVK precisa também dobrar esses fios, cortá-los e ao término da operação de inserção dobrá-los novamente pela parte inferior da placa para estabilizar o componente até que este seja soldado em uma fase posterior do processo produtivo.

Os blocos principais da insersora AVK são: uma única cabeça de inserção de componentes, um mandril que se move entre o ponto de retirada de componentes (fixo na máquina) e o ponto de inserção (na área da placa) transportando o componente do escaninho para a cabeça de inserção, dois conjuntos de escaninhos para alojar os componentes e uma mesa XY para movimentar a placa de circuito impresso sob a cabeça de inserção conforme mostra a figura 2. Uma unidade de controle comanda todo o conjunto, interpretando o programa de inserção para cada placa. Essa unidade de controle tem capacidade para 8 programas e uma memória total de 2000 posições. Assim pode-se ter desde um único programa com 2000 inserções até 8 programas cuja soma não exceda a memória total. Outrossim, caso já estejam armazenados oito programas na memória, o restante da memória fica indisponível mesmo que a quantidade de memória por eles ocupada não exceda a memória total.

Os componentes eletrônicos a inserir podem ser alojados em carretéis

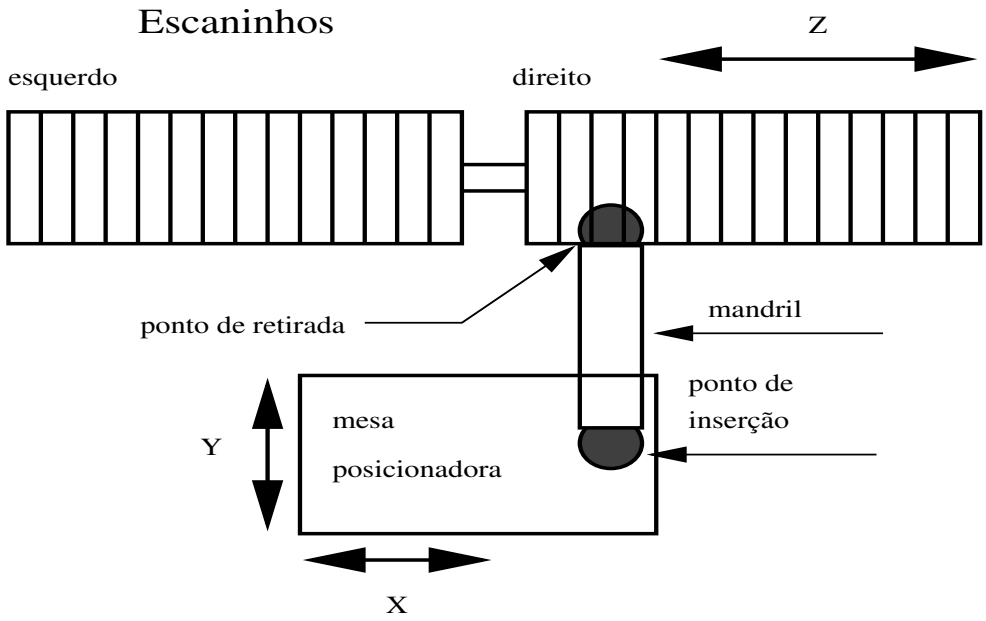

Figura 2: Blocos Principais da Insersora AVK

(considerado forma opcional pelo fabricante) ou em dispensadores feitos de caixas de papelão (porta-fitas), denominados de *flat pack*. Neste último caso, os componentes são dispostos de forma equidistante em fitas de papel ou material plástico flexível e acondicionados nessas caixas de forma similar a formulários contínuos, com as fitas dobradas em "Z" ao longo do eixo longitudinal das caixas[9]. Essas caixas por sua vez são alojadas em escaninhos providos na AVK. As fitas de componentes saindo desses escaninhos são conduzidas por guias de alumínio fazendo que elas façam um giro de 90 graus até atingir certas fendas que sujeitam as fitas de componentes para que o mandril possa sacar os componentes quando estiver alinhado com a fenda, isto é, esse determinado componente é aquele pronto para inserção.

Os escaninhos estão divididos em dois grupos, um denominado *esquer-*

[9]Cada caixa contém 2000 componentes quando cheia.

do "ZL" e outro denominado *direito* "ZR", e se movimentam por meio de servomotores. Cada um destes grupos contém 60 fendas distintas.

A AVK pode receber fitas de componentes tendo duas larguras: 26 mm ou 52 mm. Quando trabalha com componentes de 52 mm, cada quatro porta-fitas de componentes ocupam seis fendas de escaninhos para acomodar a largura maior. Neste caso, o operador da máquina pode escolher livremente qual(is) fenda(s) que deverá(ão) ficar vaga(s). Dessa forma, no limite cada grupo pode operar ou com 40 componentes nos escaninhos ocupados com componentes de 52 mm (80 ao todo) ou com 60 componentes nos escaninhos ocupados com componentes de 26 mm (120 ao todo).

Os dois conjuntos de escaninhos podem ser usados das seguintes maneiras:

- **Modo Troca** Os mesmos componentes são colocados no mesmo arranjo tanto no conjunto esquerdo como no conjunto direito de escaninhos. Um dos conjuntos é posto em operação, enquanto o outro aguarda em uma posição de espera. Quando os componentes acabam no conjunto ativo, a máquina transfere para o conjunto em espera a alimentação de componentes para inserção. Nessas condições pode-se proceder com a recarga de componentes no(s) escaninho(s) esgotado(s). O primeiro conjunto será ativado novamente somente quando os componentes do segundo conjunto terminarem.

- **Modo Troca Prioritária** Os mesmos componentes no mesmo arranjo são colocados nos dois conjuntos de escaninhos. Um dos conjuntos é designado como prioritário e o outro como secundário. Quando os componentes acabam no conjunto prioritário, a máquina segue a operação, passando para o conjunto secundário. Quando os componentes tenham

sido repostos no conjunto prioritário, a máquina reverte para ele para continuar a produção, sem que se tenham necessariamente esgotado os componentes do conjunto secundário.

- **Modo de Preparação** Um conjunto tem os componentes arranjados para uma determinada produção, enquanto o outro possui arranjo diverso, apropriado a outra placa, por exemplo. Essa disposição permite ao fabricante fazer uma troca rápida na sua linha de produção, especialmente quando os lotes a fabricar são pequenos.

- **Modo Conectado** Neste modo os dois conjuntos são interligados mecanicamente, podendo operar como um porta escaninhos com capacidade para até 120 tipos de componentes.

O movimento da placa de circuito impresso é realizado pela mesa XY com movimentos independentes e simultâneos nos dois eixos.

Usando a classificação desenvolvida na seção 2.3, esta máquina é uma máquina *concorrente*, pois mais de uma operação fundamental pode ser executada ao mesmo tempo.

Na AVK, o posicionamento do conjunto de escaninhos e o movimento da mesa XY (e nesta em cada eixo independentemente) são operações concorrentes. Por outro lado, as operações de retirada e transporte do componente e a inserção são aquelas que ultimadamente dão a *cadência* da máquina; em particular, a AVK pode ser adquirida em duas versões, uma com cadência de 5 componentes por segundo e outra com 2,5 componentes por segundo. Essa cadência é definida pelo fabricante para as seguintes condições ditas normais: a distância (em X ou Y) da posição anterior para a próxima inserção seja

menor ou igual a 30 mm, a diferença de medida de largura de dobra dos fios do componentes seja menor ou igual a 5 mm e a troca de escaninho não ultrapasse um adjacente.

O fato da mesa XY ter seus eixos independentes implica que para um dado deslocamento, o tempo gasto nesse deslocamento será determinado pela maior distância (X ou Y), sendo portanto dado pela *distância de Chebychev*, dada por $\max(|x_2 - x_1|, |y_2 - y_1|)$.

Quando a AVK é utilizada no modo conectado, os escaninhos que se localizam no limite de cada grupo ficam mais afastados que a distância inter-escaninhos de um mesmo grupo da máquina, criando uma descontinuidade no custo de troca de escaninhos (tempo) nessa posição particular.

A máquina insersora AVK executa, para efetuar uma inserção, as seguintes operações fundamentais:

[1] posicionamento do conjunto de escaninhos;

[2] retirada do componente do escaninho corrente;

[3] transporte do componente desde o alimentador até a cabeça de inserção;

[4] posicionamento da placa de circuito impresso sob a cabeça de inserção;

[5] inserção do componente na placa de circuito impresso.

Note-se a correspondência destas operações com aquelas descritas em 2.2.

Na operação [2] a fita é cortada, liberando o componente do escaninho. Durante o transporte do componente na operação [3], os fios do componente

são dobrados na distância correta para os furos na posição de inserção, girados na orientação correta, e caso a máquina esteja em um modo especial de trabalho, na operação [4], executa um refinamento no posicionamento da cabeça de inserção através de uma câmara de vídeo. Na operação [5], a finalização da inserção é executada pelo corte dos fios do componente na parte inferior da placa.

A operação [1] é uma operação onerosa em termos de tempo em relação aos outros movimentos da máquina, pois envolve a movimentação da massa de todo o conjunto de escaninhos com os componentes ali alojados.

Claramente, mesmo que algumas operações sejam executadas de forma concorrente, há uma certa precedência entre elas. A operação [4] pode ser realizada em paralelo com as operações [1], [2], ou [3], devendo entretanto preceder a operação [5]. Essas relações são melhor mostradas no grafo orientado da figura 3.

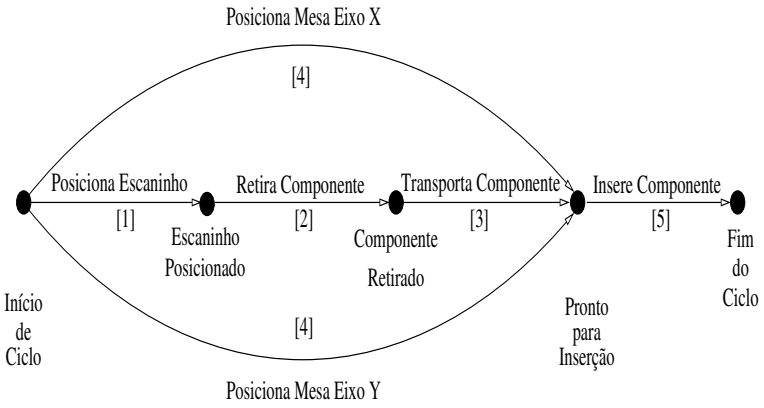

Figura 3: Relacionamento dos Eventos e Operações na AVK

Para o usuário, programar a insersora significa gerar um plano de atribuição de componentes a cada escaninho, e um programa de inserção, indi-

cando em que posição cada componente vai ser inserido[10].

2.6 Análise dos Tempos de Operação da AVK

Conhecendo as operações fundamentais da máquina insersora AVK, precisamos determinar o seu tempo de ciclo de máquina, ou seja, o tempo necessário para efetuar a inserção de um componente na placa. Vamos utilizar a seguinte notação:

m: número de tipos componentes necessários;

n: o número total de componentes a ser inseridos;

i: índice para os ciclos da máquina;

c_1: tempo de cadência da máquina (constante que indica o tempo mínimo de ciclo);

c_2: tempo de carga de uma placa na mesa XY (constante);

$x(i)$: tempo para deslocar a mesa XY da posição x_{i-1} para posição x_i;

$y(i)$: tempo para deslocar a mesa XY da posição y_{i-1} para posição y_i;

$z(i)$: tempo para deslocar o conjunto de escaninhos da posição z_{i-1} para posição z_i;

$w(i)$: tempo para modificar a abertura da pinça que dobra os fios dos

[10]Deve incluir também a largura da dobra, a altura de inserção, o giro (em múltiplos de 90 graus, apenas) e se necessário um *offset* para ângulos não múltiplos de 90 graus.

componentes da largura w_{i-1} para a largura w_1;

O tempo de duração de um ciclo i vale:

$$\tau(i) = \{\max[c_1, x(i), y(i), z(i), w(i)]\} \tag{1}$$

O tempo total de inserção de uma placa será então:

$$T_o = \sum_{i=1}^{n} \tau(i) + c_2 \tag{2}$$

O tempo de inserção e o deslocamento da cabeça do ponto de retirada dos componentes até o ponto de inserção são tão pequenos frente aos outros tempos que ficam absorvidos no tempo c_1.

Surprendentemente, a documentação fornecida ao usuário dessas máquinas insersoras não traz informações suficientes para estabelecer os tempos das operações fundamentais! Esse problema é também comentado por McGinnis et al. [MCG92], que afirma ter havido poucos resultados publicados que apresentem confirmação experimental dos modelos de tempo de ciclo. Crama et al. [CRA90] citam ainda diferenças de descrição de operação para uma mesma máquina (Fuji CP-IV).

Para sintonizar as constantes do modelo da AVK, utilizou-se neste trabalho dados de programas de inserção especialmente gerados para levantar os tempos principais da insersora, mantendo fixas as outras operações. Estes dados foram depois validados comparando os tempos obtidos pela simulação da inserção de programas de placas de produção com os tempos reais obtidos

na máquina.

3 OTIMIZAÇÃO

Neste capítulo, abordamos o problema de inserção automática de componentes eletrônicos sob o prisma de um problema de otimização. Primeiramente, vamos tratá-lo de uma forma mais qualitativa, mostrando que ele se decompõe em dois sub-problemas mais gerais de otimização, o TSP e o QAP, e que esses dois problemas são imbricados, exceto em casos especiais. Tais problemas gerais são então apresentados, bem como alguns algoritmos que os resolvem. Finalmente, formulamos matematicamente o problema de otimização a ser solucionado.

3.1 Análise Qualitativa do Problema de Otimização de Inserção

Analisando-se a operação da insersora e a equação de tempo total de inserção de uma placa de circuito impresso como exemplificado na equação (2), fica claro que a **seqüência** de inserção influi no tempo total de inserção, uma vez que uma escolha infeliz pode levar a mesa posicionadora a fazer grandes movimentos entre inserções seguidas, onerando o tempo total de inserção. Por outro lado, vê-se que a **atribuição** dos tipos de componentes aos escaninhos para uma dada seqüência de inserção também influi no tempo total, pois se entre duas inserções o escaninho anterior àquele contendo o componente a inserir estiver muito afastado deste último, a insersora terá seu tempo de inserção dominado pelo tempo de troca de escaninhos. Como já observamos no capítulo 2, dado que o conjunto de escaninhos é o bloco (móvel) com maior massa na insersora, é de se esperar que esse movimento da máquina

seja mais lento.

Esse fato é relativamente fácil de se constatar observando a operação da insersora, o que levou os operadores de uma instalação a estabelecer uma estratégia extremamente simples: para cada escaninho o programa executa todas as inserções daquele componente na placa, só então passando ao escaninho seguinte [11]. Esta estratégia decorre do fato de que, na maior parte dos casos, componentes similares são inseridos em regiões próximas (por exemplo, diodos de potência são geralmente colocados próximos uns dos outros para formar o circuito retificador da fonte). É interessante observar que esse parece ser o comportamento natural quando as soluções são geradas manualmente, uma vez que Broad et al. [BRO96] citam uma experiência similar, e a instalação sob estudo localizava-se na Nova Zelândia!

Essa estratégia entretanto, não garante o menor tempo de inserção. Supondo que a regra mencionada acima não seja válida, certos tipos de componente deverão ser inseridos em vários pontos esparsos e afastados na placa de circuito impresso, e como conseqüência a mesa efetuará diversos movimentos de vai e vem a mesmas regiões da placa para inserir os diversos componentes; a única condição que faria essa abordagem apropriada seria o tempo de troca dos escaninhos ser da ordem do tempo de percurso na placa para inserção de um determinado tipo de componente[12]. Melhor seria aproveitar o tempo necessário para um deslocamento grande da mesa XY e ter os componentes atribuídos nessas duas posições em dois escaninhos mais afastados, de sor-

[11] Rigo, A. L. (IPT, São Paulo). "Comunicação Pessoal",1997.

[12] Esse é o caso, por exemplo, de certas máquinas ferramenta, como as furadeiras automáticas de placas de circuito impresso, onde para trocar de broca o mandril necessita deslocar-se para uma posição de origem e a distribuição dos furos dos vários diâmetros é bem espalhada pela placa [REI94].

te que as inserções seguintes utilizassem os escaninhos mais adequados para suas posições, aproveitando o tempo do deslocamento grande da mesa para fazer o salto de escaninhos.

Portanto, o que se deve buscar, em última análise, é o melhor "casamento" entre os tempos obrigatórios devidos aos deslocamentos da mesa posicionadora com os tempos obrigatórios devidos às trocas de escaninhos, de modo a se ter as operações concorrentes com tempos os mais próximos possíveis.

Nota-se também que a escolha da melhor atribuição dos componentes aos escaninhos é condicionada pela seqüência de inserção na placa, e por outro lado a melhor seqüência de inserção pode ser diferente do mínimo percurso (planar) na placa entre os pontos de inserção devido ao custo de troca de escaninhos (figura 2). A melhor seqüência de inserção é condicionada portanto tanto pela atribuição dos componentes aos escaninhos, quanto pela seqüência de inserção.

A determinação do menor percurso, ou seja, da melhor seqüência de inserção levando em conta os custos (tempos) dos ciclos, conforme a equação (1), pode ser formulada como um problema do tipo do TSP, enquanto a atribuição dos componentes aos escaninhos que minimize o tempo total de inserção pode ser formulada como um problema do tipo do QAP. Nas seções 3.2 e 3.3, analisamos esses problemas e suas características.

Uma primeira pergunta que surge então ao se tentar resolver o problema de otimização da inserção de componentes na máquina AVK é se existe alguma forma de desacoplar esses dois problemas, eventualmente criando uma hierarquia entre eles, caso contrário qualquer abordagem de solução envolveria a necessidade de iterações para resolvê-lo.

Considerando m o número de tipos componentes necessários para montar uma determinada placa, M o número de escaninhos disponíveis e n o número de componentes a inserir nessa placa (onde nessas n inserções pode haver ou não repetição da inserção de um ou mais dos m tipos de componentes), podemos considerar os seguintes casos:

n qualquer, $m = 1$: neste caso obviamente não há nenhuma restrição quanto à atribuição do porta-fitas desse componente em nenhum escaninho e o problema torna-se um *TSP puro*, sujeito aos custos dos ciclos da equação (1), ficando os tempos dominantes os devidos às distâncias XY e os tempos de mudança de largura de dobra dos componentes[13].

$m \leq M, n = m$: neste caso temos uma única inserção de cada tipo de componente. A atribuição de cada componente aos escaninhos pode ser sempre decidida *após* a determinação da melhor seqüência de inserção, resolvendo-se o TSP, e levando em conta neste caso um custo fixo de uma troca obrigatória de escaninho a cada inserção na equação (1). Em outras palavras, temos uma solução trivial para o QAP, que corresponde à seqüência de inserções encontrada, fazendo corresponder o primeiro componente a ser inserido ao primeiro escaninho, o segundo componente ao segundo escaninho e assim por diante.

[13]Este caso ocorre quando a AVK é fornecida com uma unidade adicional para instalação de *jumpers*. A alimentação para fazer esses *jumpers* é um rolo de fio de cobre estanhado. A otimização do tempo de inserção dos *jumpers* resume-se a encontrar o percurso mais curto.

$\boldsymbol{n > m}$**:** neste caso há claramente repetição de tipos de componentes para pelo menos algumas posições de inserção, podendo-se examinar dois subcasos:

$\boldsymbol{m < M}$**:** esta situação propicia uma folga de $M - m$ escaninhos vagos aos quais se pode atribuir alguns componentes repetidos que tenham alguma característica relevante no projeto da placa, do ponto de vista do processo de inserção automática (maior número de repetições, distância máxima em que comparecem na placa). Deste modo, pode-se posicionar tais componentes estrategicamente em um desses escaninhos de tal forma a servir uma certa área da placa, e com isto evitar a necessidade de saltos de escaninhos. Tal estratégia é mais adequada quando não houver nenhuma distância equivalente em tempo que se possa "casar" o custo com esse salto. Quando se aborda o problema de atribuição dos componentes aos escaninhos com mais esse grau de liberdade, surge uma nova variável a explorar na otimização, ou seja, decidir de *qual* de dois (ou mais) escaninhos deve-se retirar o componente apropriado para uma dada posição. Esse problema adicional, chamado de *recuperação* mostra-se ser também $\mathcal{NP}$-difícil, exceto para casos específicos e é endereçado por Crama et al. [CRA96b].

$\boldsymbol{m = M}$**:** nesta situação, o porta escaninhos não oferece nenhuma folga para se atribuir uma repetição de algum tipo de componente. Este caso é o mais geral e no qual o TSP e o QAP são mais interdependentes, e exige no mínimo uma abordagem interativa para sua formulação e solução.

***m > M*:** neste caso temos que particionar o problema em duas ou mais passadas de inserção, com $m_1, m_2, \ldots, m_n \leq M$. A decisão de *como* fazer esse particionamento faz com que o problema deixe de ser do tipo de Arranjo e Seqüenciamento (nível 3) e passe a ser um problema de Alocação[14] (nível 2), conforme descrito na seção 2.4. Esse problema é endereçado por Maimon e Shtub [MAI91], e Ben-Arieh e Dror [ARI90].

Em todos os casos acima, quando indicamos n qualquer queremos dizer até os limites da máquina, ou mais especificamente pela capacidade de memória da sua unidade de controle.

Estes casos estão resumidos na tabela 1 a seguir.

Caso	$m = 1$	$1 < m < M$	$m = M$	$m > M$
$n = m$	TSP puro	TSP + QAP trivial	TSP + QAP trivial	Alocação
$n > m$	TSP puro	TSP + QAP + recuperação	**TSP + QAP**	Alocação

Tabela 1: Taxionomia de Casos na Otimização da AVK

O caso em negrito na tabela acima é o problema de otimização que vamos tratar neste trabalho.

[14] Pode-se inclusive chegar ao nível 1, se houver máquinas que suportem mais de uma tecnologia e que seja necessário estudar o compromisso de particionamento do problema, por exemplo, uma máquina mais lenta que outra, etc.

Embora os casos especiais acima sejam os únicos em que se possa desacoplar o TSP do QAP (havendo até o caso limite em que não é necessário formular, nem resolver o QAP), eles podem servir para gerar problemas de inserção específicos para verificar, quase que por simples inspeção, se o sistema desenvolvido está conseguindo encontrar boas soluções, principalmente no que diz respeito aos algoritmos de QAP.

3.2 O Problema do Caixeiro Viajante ("TSP")

O "Problema do Caixeiro Viajante" ou em inglês *Traveling Salesman Problem* (TSP) pode ser apresentado na sua forma mais simples como segue:

"Um vendedor necessita visitar n cidades, sendo permitida uma única visita a cada uma delas, iniciando o percurso a partir de uma cidade qualquer e retornando ao lugar de partida. Qual rota, ou percurso, deve ser escolhida para minimizar a distância total viajada?"

Em vez de distância, outras noções alternativas como tempo, custo, etc. podem ser consideradas.

Matematicamente, o problema pode ser formulado da seguinte maneira:

Dada uma *matriz de custo* $D = (d_{ij})$, onde d_{ij} = custo de percurso da cidade i para cidade j, $(i, j = 1, 2, \ldots, n)$, encontre uma permutação $P = (i_1, i_2, i_3, \ldots, i_n)$ de inteiros de 1 até n que minimize a quantidade:

$$T = d_{i_1 i_2} + d_{i_2 i_3} + \ldots + d_{i_n i_1} \tag{3}$$

Essa formulação mostra o carácter combinatório do TSP.

Alternativamente pode-se propor uma formulação usando programação linear. Neste caso, o problema seria formulado como:

Condiderando a mesma *matriz de custo* D, e introduzindo as variáveis de decisão x_{ij} pede-se para minimizar a quantidade:

$$T = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{4}$$

sujeita às condições:

$$x_{ii} = 0, \quad i = 1, \ldots, n \tag{5}$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n \tag{6}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n \tag{7}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n \tag{8}$$

e para evitar que se construam percursos parciais, acrescentam-se as seguintes restrições para qualquer subconjunto $S = \{i_1, i_2, \ldots, i_r\}$ de inteiros de 1 a n:

$$x_{i_1 i_2} + x_{i_2 i_3} + \ldots + x_{i_{r-1} i_r} + x_{i_r i_1} \begin{cases} < r & \text{para} \quad r < n \\ \leq n & \text{para} \quad r = n. \end{cases} \tag{9}$$

Nessa formulação gera-se um número elevado de restrições para evitar percursos parciais mesmo para valores modestos de n (mais exatamente $2^n - 2$).

Por essa razão, abordagens para resolver o TSP empregando programação linear buscam trabalhar inicialmente apenas com parte das restrições, fazendo relaxações que simplifiquem a formulação (embora ampliando o espaço de soluções exeqüíveis) para em seguida re-introduzir restrições até que as soluções satisfaçam ao problema completo [RAR98] [LAW85]. O problema formulado exclusivamente pelas equações (5) a (8) denomina-se *problema de atribuição* (*assignment problem*). Trata-se de uma formulação de programação linear inteira, devido à função objetivo ser linear e ao fato das variáveis de decisão serem inteiras. É um problema de conhecida solução, vide por exemplo Taha [TAH97].

Historicamente, a primeira abordagem para solucionar o TSP empregou essa formulação dada por (5) a (9), no artigo que é considerado um dos mais significativos eventos na história da otimização combinatória, denominado "Solution of a large-scale traveling-salesman problem" por Dantzig, Fulkerton e Johnson, em 1954 e publicado no *Journal of Operations Research Society of America* [DAN54]. Contudo o número elevado de restrições dado por (9) levou à proposta de Miller, Tucker e Zemlin em 1960 [MIL60][GOU99].

Designando arbitrariamente uma cidade como a origem, com u_i e u_j números reais arbitrários, as restrições a seguir impõem a mesma proibição de ausência de percursos parciais:

$$u_i - u_j + nx_{ij} \leq n - 1, \quad i, j = 2, \ldots, n, \tag{10}$$

Com essas restrições, a formulação passa a ser de programação inteira mista, uma vez que temos as variáveis de decisão x_{ij} binárias (inteiras) e

$n-1$ variáveis contínuas. Contudo o fato mais notável dessa formulação é que a equação (10) representa apenas $(n-1)^2$ restrições, ou seja, temos agora um modelamento utilizando $O(n^2)$ restrições ao invés de $O(2^n)$ dado pela equação (9).

O TSP é um problema clássico, e prova-se a sua $\mathcal{NP}$-*completude* [LEW81], pois os algoritmos rigorosos para solucionar esse problema requerem um tempo exponencial em função do número de cidades a visitar. Colocado de outra forma, o único algoritmo que se conhece que resolve **todas** as instâncias do TSP é a completa enumeração de todas permutações cíclicas P, avaliando o comprimento (custo) total de cada uma delas e selecionando a melhor. A ordem do tempo necessário para rodar esse algoritmo é $O(n!)$.

O TSP é *simétrico* se a distância (custo) entre um nó i e um nó j é a mesma de j para i (ou seja $d_{ij} = d_{ji}$ para $\forall i, j \in 1, 2, \ldots, n$) e diz-se satisfazer a desigualdade triangular se $d_{ik} \leq d_{ij} + d_{jk}$ para três i, j, k quaisquer onde d_{ab} é a distância entre a e b [LAW85].

A simplicidade da proposição do TSP esconde à primeira vista a dificuldade de sua solução. Na verdade, ele é ainda um desafio suficiente para lhe render um sítio (*site*) na Internet (`http://www.ing.unlp.edu.ar/cetad/mos`) dedicado a concentrar os esforços na solução de instâncias do TSP e partilhar a experiência internacional nesse campo. Os resultados têm progredido da solução conhecida para vinte cidades [LIN65] ao estado atual onde se conhece a solução para problemas de até 2000 cidades como citado por [KAN97][15].

A seguir, apresentamos alguns algoritmos que resolvem este problema.

[15] [REI94] menciona comunicações pessoais de soluções para 3038 e 4461 cidades, obtidas por Applegate, Bixby, Chvàtal e Cook, respectivamente em 1991 e 1993.

3.2.1 Lin Kernighan

Este algoritmo, originalmente desenvolvido por Lin e Kernighan [LIN73], baseia-se na idéia de melhorar percursos pela troca de arcos, de modo a obter-se um percurso mais curto. A primeira aparição dessa técnica trocando dois arcos de cada vez ocorreu no artigo de Croes [CRO58] e trocando três arcos de cada vez no artigo de Lin [LIN65]. Procedimentos que empregam a troca de arcos costumam ser chamados de *r-opt* para uma troca de r arcos, sendo então o algoritmo de Lin um 3-opt e o de Croes um 2-opt. No caso de uma troca 2-opt, conforme a figura 4, dois arcos são desconectados de quatro cidades e essas cidades são reconectadas por dois novos arcos. Caso essa nova configuração resultar em um percurso mais curto, esse movimento é mantido, caso contrário retorna-se à situação anterior. No caso 3-opt, como mostra a figura 5, três arcos são desconectados e as seis cidades são reconectadas com três novos arcos, novamente buscando um percurso mais curto. É interessante observar que no caso 2-opt existe apenas uma outra maneira de reconectar os arcos (desprezando a reconexão que restitui a configuração), enquanto que no caso do 3-opt têm-se oito maneiras de reconectar os três arcos. A verificação da existência de um movimento 2-opt que melhora um percurso tem uma complexidade $O(n^2)$, uma vez que se tem de considerar todos os pares de arcos, enquanto que a verificação de uma melhoria de um movimento 3-opt tem uma complexidade de $O(n^3)$.

Embora o tempo de execução de um 3-opt aumente em relação a um 2-opt, seu desempenho, em termos da qualidade das soluções que obtém, também melhora. Contudo, para problemas com números de cidades grandes, o tempo

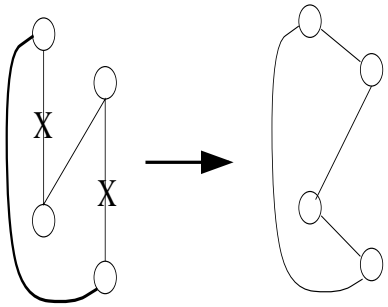

Figura 4: Um Movimento 2-opt

de máquina necessário pode ficar proibitivo, mesmo sendo polinomial[16].

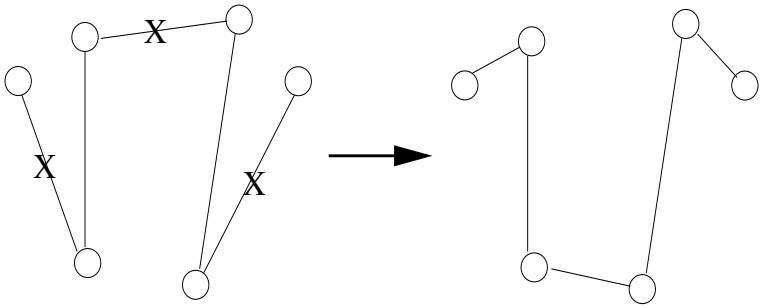

Figura 5: Um Movimento 3-opt

A idéia do algoritmo de Lin e Kernighan é o uso de um procedimento r-opt onde r é variável, decidido de forma dinâmica a cada iteração do algoritmo. Uma vez que se tenha encontrado um movimento envolvendo s arcos, uma série de testes determina se há alguma melhoria possível movimentando $s+1$ arcos, continuando-se com esse procedimento até que condições de parada sejam atingidas. Note-se que durante todas esses movimentos, sempre estão se formando percursos válidos, ou seja, não se deixam pendentes arcos ou cidades em movimentos temporários para que depois tenha que se retroceder

[16][REI94] cita tempo de 3 horas para um problema de 654 cidades usando 3-opt.

alguns passos no algoritmo.

Algoritmo lin-kernighan

[1] ($T = T_0$) Seja T um percurso inicial

[2] ($p = 1$) Selecione uma cidade qualquer como inicial, e considere um dos arcos do percurso atual adjacente a esta cidade para retirada.

[3] Desde a outra ponta deste arco, escolha um arco que não pertença ao percurso corrente de forma a minimizar o percurso.

[4] ($p = p + 1$) Escolhido o arco a ser retirado, que deve ser um dos dois arcos adjacentes à cidade escolhida no passo [2] para que as cidades restantes permaneçam conectadas, o arco a ser adicionado fica unicamente determinado, e sua adição deve reconstituir o percurso.

[5] O arco encontrado no passo [4] pode ser substituído por um mais de um arco caso se encontre uma nova cidade de forma que estes novos arcos diminuam ainda mais o percurso atual. Incrementa-se p e permanece-se neste processo até que

  [5.1] não se encontre mais nenhuma troca viável,

  [5.2] a configuração presente seja um percurso,

  [5.3] nenhuma troca viável reduza o percurso.

[6] Repita desde o passo [2] para cada cidade até todas as cidades tenham sido usadas e nenhuma melhora adicional tenha sido obtida.

Fim lin-kernighan

3.2.2 *Greedy* Constructive Search

De modo geral, algoritmos *greedy* (também denominados "míopes") são aqueles que constrõem uma solução do seguinte modo: em cada passo, elegem a

próxima variável a ser fixada e o seu valor de tal forma que cause o mínimo impacto na viabilidade da solução e se obtenha o melhor ganho na função objetivo, baseado no ganho da solução parcial corrente [RAR98]. Tais algoritmos trabalham somente com informações locais ao ponto de busca, daí seu nome, que significa algo como "guloso".

No TSP, os algoritmos podem ser estudados como uma variação da heurística da economia, desenvolvida originariamente para resolver problemas de roteamento de veículos [CLA64]. Para tratar essa classe de problemas como um TSP, considera-se a existência de um único veículo (ou seja, os problemas de roteamento de veículos têm como sub-problemas TSPs) [REI94]. A idéia da heurística *greedy* é imaginar um percurso com n caminhos de comprimento 0 e ir verificando em cada passo se o menor arco ainda não considerado (incluído no percurso) pode ser utilizado para unir duas cidades conectando dois outros arcos. O procedimento termina quando um percurso fechado passando por todas as cidades é encontrado.

Considere-se $E_n = \{e_1, e_2, \ldots, e_m\}$ o conjunto de arcos que ligam todas as n cidades duas a duas, onde $m = n(n-1)/2$, cada arco com um custo c_i associado. O algoritmo encontra-se descrito a seguir.

Algoritmo greedy

[1] Ordene E_n de forma a ter $c_1 \leq c_2 \leq \ldots, \leq c_m$

[2] $T = \emptyset$

[3] Para $i = 1, 2, \ldots, m$:

[3.1] Se $T \cup \{e_i\}$ pode ser estendido em ciclo fechado, então faça $T = T \cup \{e_i\}$.

Fim greedy

Para TSPs satifazendo a desigualdade triangular, este algoritmo pode chegar a percursos $\log n$ vezes o comprimento do percurso ótimo [FRI79]. Considerando que a etapa de ordenação [3.1] possa ser executada em tempo constante (atingível pelo uso de estruturas de dados apropriadas), chega-se ao tempo $O(n^2 \log n)$ [REI94].

3.3 O Problema da Atribuição Quadrática ("QAP")

O Problema da Atribuição Quadrática ou em inglês *Quadratic Assignment Problem* (QAP) é um problema que tem sua formulação original dada por Koopmans e Beckman em 1957 num contexto de determinar a melhor alocação de usinas (plantas industriais) a localidades [KOO57]. O problema consistia em alocar n usinas indivisíveis a n posições de forma a maximizar a renda total dessas usinas, através da minimização do custo de transporte de mercadorias entre usinas. Neste problema temos duas matrizes, uma denominada *matriz de fluxo* entre as usinas $F = (f_{ij})$, indicando o número de unidades de uma mercadoria a ser transportada entre a i-ésima e a j-ésima usina, e a outra denominada *matriz das distâncias* entre as localidades $D = (d_{kl})$, indicando a distância entre a k-ésima e a l-ésima posições.

Matematicamente, esse problema pode ser formulado como segue:

Seja uma *matriz de custo* $C = (c_{ijkl})$, onde c_{ijkl} = custo do par de usinas (i, j) no par de posições (k, l). Usando as matrizes definidas acima, um elemento de C seria escrito como: $c_{ijkl} = f_{ij} d_{kl}$.

Seja ainda $\rho(i)$ uma função que para uma determinada planta i retorna a sua localidade. Encontre uma permutação $\rho = (\rho(1), \rho(2), \ldots, \rho(n))$ dos inteiros $1, 2, \ldots, n$ que minimize a quantidade:

$$T = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij\rho(i)\rho(j)} \tag{11}$$

Essa formulação também enfatiza o caráter combinatório desta classe de problemas.

Formulando o problema com programação inteira, ele passa a ser expresso da seguinte maneira:

Encontre o valor das variáveis $x_{ij}, i, j = 1, 2, \ldots, n$, que minimizem a quantidade:

$$T = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} c_{ijkl} x_{ij} x_{kl} \tag{12}$$

sujeito às condições:

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n \tag{13}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n \tag{14}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n \tag{15}$$

As somatórias (14) e (15) impõem que toda usina seja alocada a uma única localidade e que toda localidade receba uma única usina.

Um QAP é denominado *euclidiano* quando a matriz A é simétrica e sua matriz de distâncias satisfaz a desigualdade triangular $d_{ik} \leq d_{ij} + d_{jk}$ para três localidades i, j, k quaisquer onde d_{ab} é a distância entre a e b [GAN94].

O que torna o QAP único em relação a problemas de atribuição é que não se pode determinar o impacto (custo) de uma atribuição até que todas as unidades tenham sido alocadas.

O QAP também tem demonstrações de sua $\mathcal{NP}$-*completude*, especialmente quando formulado como um problema de decisão [SAH76] ou através de uma demonstração por Lawler [LAW85], onde mostra-se que o TSP pode ser transformado em um QAP em tempo polinomial.

Para QAPs, a otimalidade das soluções está ainda em um patamar mais baixo, sendo que o número de localidades está no redor de 20 a 30 [LAC93].

A seguir, apresentamos alguns algoritmos que resolvem este problema.

3.3.1 Algoritmo Baseado no Húngaro

O algoritmo húngaro não é propriamente um algoritmo para a solução do QAP, mas sim um algoritmo para a solução de um problema mais simples denominado *Linear Assignment Problem*[17].

O LAP é formulado da seguinte maneira: Dada uma *matriz de custo* $C = (c_{ij})$ quadrada de ordem n, determinar uma *matriz de solução* $X = (x_{ij})$ de forma a minimizar a quantidade:

[17] Por concisão, doravante utilizaremos apenas a abreviação LAP para fazer menção a esse tipo de problema.

$$T = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{16}$$

sujeito às condições:

$$x_{ii} = 0, \quad i = 1, \ldots, n \tag{17}$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n \tag{18}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n \tag{19}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n \tag{20}$$

A matriz C indica o custo[18] de alocar um determinado i recurso à posição j. Por exemplo, no caso de decisões do nível 2 (alocação) descritas na seção 2.4, considerando-se o custo de operação de n máquinas executando n operações, a resolução do LAP encontra o custo total mínimo, determinando para cada máquina qual deve ser o serviço a ser alocado (operação).

O LAP descrito pelas equações (16) a (18) pode ser visto como uma relaxação do TSP, uma vez que essas equações têm formulação similar às apresentadas nas equações (4) até (8). As restrições adicionais, presentes no TSP e ausentes no LAP, servem para evitar os percursos parciais (formação de laços desconectados).

A diferença principal entre o LAP e o QAP é que no primeiro pode-se determinar o impacto da alocação de um certo recurso a um serviço sem

[18]Pode-se também representar benefícios, sendo que neste caso o problema passa a ser de maximização.

necessidade de se levar em conta as outras atribuições, enquanto no QAP só é possível avaliar esse impacto após feitas todas as alocações devido aos custos de intercâmbio entre os recursos. Em outras palavras, no caso do LAP, os custos associados à alocação de diversos recursos são independentes entre si. No nosso problema de otimização da inserção automática de componentes eletrônicos, tal algoritmo não se aplica, pois alocar um tipo de componente a um escaninho **tem** um efeito sobre a alocação alguns escaninhos vizinhos a serem utilizados (o tempo de deslocamento).

O LAP é considerado um caso especial do problema de transporte [TAH97] [SPI71]. Entretanto, dadas as suas características específicas, é mais facilmente resolvido por algoritmos especializados.

Para fazer uso deste algoritmo é necessário reduzir o QAP para um LAP. Para tal, consideramos um subconjunto de usinas:

$$\{U_p\}, \qquad 1 \leq p \leq n,$$

tal que:

$$f_{ij} = 0 \quad \forall i, j \in U_p, \quad 1 \leq p \leq n,$$

onde f_{ij} é o fluxo entre usinas como definido na seção 3.3.

Os subconjuntos U_p são independentes, no sentido de não haver duas usinas pertencentes ao mesmo subconjunto que tenham qualquer intercâmbio de material entre elas. Embora esta não seja uma condição necessária, é

conveniente utilizar apenas os conjuntos maximamente independentes, isto é, aqueles que não estejam contidos em nenhum outro conjunto independente.

Quando as usinas $u \notin U_P$ são fixadas, o QAP se reduz a um LAP, pois todos os custos de atribuição quadráticos não considerados envolvem pares de usinas $(i, j) \in U_p$, com $f_{ij} = 0$.

Desta forma, os coeficientes de custo do QAP podem ser classificados em três grupos:

(a) aqueles que têm ambas usinas alocadas (nenhuma das duas em U_p), contribuindo com um custo constante Z_a para a função objetivo;

(b) aqueles com uma usina alocada (uma em U_p, a outra não), definindo um LAP com solução ótima Z_b;

(c) aqueles com ambas usinas não alocadas (ambas pertencentes a U_p) contribuindo com um custo $Z_c = 0$ para a função objetivo.

Deve-se iterar ciclicamente os conjuntos U_p até que nenhuma melhoria seja obtida para um ciclo completo de iterações.

Algoritmo HWG

[1] Construa uma matriz C' a partir da matriz C, como segue:

[1.1] Para cada

[1.1.1] linha na matriz C identifique o menor valor e subtrai-o de todos os elementos dessa linha.

[1.1.2] coluna na matriz modificada no passo [1.1.1], identifique o menor valor e subtrai-o de todos os elementos dessa coluna.

[2] Se cada linha da matriz C' tiver um único zero, a atribuição mínima foi encontrada. Parar.

[3] Se não foi encontrada uma solução viável então:

[3.1] Marcar o mínimo número de linhas e colunas que cubram todos os zeros na matriz C'.

[3.2] Selecione o mínimo elemento não coberto por essa marcação e subtraia esse valor de todos os elementos não cobertos.

[3.3] Adicione esse valor a cada elemento que estiver na intersecção da marcação feita no passo [3.1].

[3.4] Vá para o passo [2].

Fim HWG

3.3.2 Permutação Parcial

Este algoritmo, também denominado CRAFT [ARM63], é relativamente simples, utilizando um método de busca local para a solução do QAP.

Partindo de uma alocação completa, isto é, com todas as usinas alocadas a uma localidade, e seu custo corrente, uma medida da custo de cada um dos $n(n-1)/2$ pares de usinas é calculado. Todas as trocas desses pares são então verificadas, uma de cada vez, e aquela que produz a maior redução no custo total é efetuada, com as duas usinas envolvidas trocando de posição para a próxima iteração. Se nenhuma troca pode produzir uma melhoria no custo, o algoritmo pára.

Algoritmo permutação parcial

[1] Calcule os custos entre os $n(n-1)/2$ pares de usinas.

[2] Para os $n(n-1)/2$ pares avalie o impacto no custo total devido à permutação do par de usinas. Selecione a permutação que causar a maior redução no custo total. Vá para 1.

[3] Se não houver nenhuma permutação que reduza o custo total, pare.

Fim permutação parcial

Algoritmos de permutação parcial, em especial o CRAFT, embora não sejam rigorosos, mostram-se capazes de encontrar soluções ótimas em várias avaliações. Em [GOL89], apresentam-se soluções com um tempo de execução $O(n^2)$.

3.4 Algoritmos para Otimização Discreta

Nas seções 3.2 e 3.3, analisamos o TSP e o QAP, verificando que trata-se de problemas de otimização combinatória e que, portanto, precluem a tratabilidade elegante dos problemas de otimização contínua. Não se pode, por exemplo, empregar os métodos do cálculo diferencial e fazer derivadas iguais a zero, pois estamos em uma situação combinatória e nosso espaço de soluções não é contínuo mas sim um conjunto de percursos (TSP) ou atribuições (QAP), ambos discretos, sendo que os problemas envolvem decisões lógicas que não podem ser modeladas apropriadamente como contínuas.

Por outro lado, nas descrições apresentadas nas seções 3.2 e 3.3 vemos que a única forma garantida de encontrar a solução desses problemas seria a enumeração total das alternativas. Essa forma de solucionar esses problemas obviamente pode ser considerada apenas para problemas de tamanho

pequeno. No nosso caso, onde a insersora AVK permite inserir até 2000 componentes por placa (equivalendo às cidades do enunciado do TSP), teríamos 2000! ($\approx 3,3 * 10^{5735}$) caminhos possíveis para pesquisar, e dispondo de 120 escaninhos 120! ($\approx 6,7 * 10^{198}$) atribuições de tipos de componentes aos escaninhos. Obviamente, essa abordagem não é prática para um número de casos tão grande e necessitamos portanto recorrer a outras estratégias de solução.

Uma vez que se reconhece a $\mathcal{NP}$-dificuldade desses problemas, devemos buscar uma forma alternativa de encontrar as soluções desejadas. Basicamente, podemos dizer que existem os seguintes grupos de métodos para a resolução de problemas de otimização discreta [VIA98]:

Exatos São algoritmos enumerativos, que varrem o espaço de soluções e buscam aquela que apresenta a melhor métrica. Distinguem-se quanto à a abrangência em:

enumeração explícita onde todo o espaço de soluções é examinado. Devido à característica de explosão combinatória dos problemas, esta abordagem só serve para resolver instâncias pequenas dos problemas.

enumeração implícita onde apenas parte do espaço de soluções é examinado. Tal espaço é particionado, de modo a eliminar conjuntos inteiros de soluções, através do uso de limites (*bounds*) obtidos por relaxações, ou pelo uso de programação dinâmica.

Aproximados São algoritmos onde não há garantia de encontrar a solução ótima, porém têm o atrativo de rodarem em tempo polinomial. Também denominados heurísticos, buscam resolver o problema base-

ados em regras empíricas, cuja aplicação costuma ser dependente do tipo de problema. Dividem-se em:

de construção onde uma solução é gerada apenas a partir dos dados do problema.

de melhoria onde os algoritmos partem de uma solução existente e usam um movimento particular para refiná-la, otimizando assim a solução.

Métodos de Programação Linear e não Linear usam algoritmos como o Simplex, Pontos Interiores, Elipsóides, etc. Para lançar mão destes métodos, inicialmente relaxa-se o problema ignorando as restrições de integralidade dos problemas, para em seguida construir a solução usando-se um algoritmo polinomial.

Estocásticos usam aleatoriedade para tentar fugir de mínimos locais a que estão sujeitos os algoritmos heurísticos, combinada com uma regra para evitar que esses algoritmos entrem em ciclos ao tentar sucessivamente soluções locais já experimentadas.

Analógicos motivados por fenômenos observados na Natureza, resolvem os problemas sem oferecer um *insight* de como alcançaram a solução. Têm com atrativo a possibilidade de ser aplicados a problemas de estrutura pouco conhecida.

A seguir descrevemos alguns destes grupos de algoritmos, bem como métodos para sintonizá-los.

3.4.1 Algoritmos Exatos

A enumeração explícita, também denominada enumeração total, de todas as soluções de um determinado problema e a posterior escolha daquela que tenha a melhor métrica para o caso sob estudo (menor distância total no caso do TSP, menor custo no caso do QAP) é uma forma de abordagem válida. No entanto, só pode ser considerada para instâncias "pequenas" dos problemas combinatórios, pois como já vimos na seção 3.4 o tamanho do espaço de soluções dos problemas combinatórios cresce de forma exponencial [RAR98].

Uma forma mais interessante de tratar o problema, então, é fazer o que se denomina *enumeração implícita* das soluções, utilizando técnicas de relaxação e restrições. Um problema é relaxado quando se considera apenas um subconjunto das condições. Uma restrição é aplicada quando novas condições são adicionadas ao problema. Efetivamente com essa abordagem se está dividindo o conjunto de soluções viáveis em subconjuntos progressivamente menores, calculando limites na função objetivo sobre cada subconjunto, e usando esses limites para descartar certos subconjuntos de consideração ulterior. Esse procedimento segue até que cada subconjunto tenha produzido uma solução viável, ou que seja provado que não contenha nenhuma solução melhor daquela já conhecida até o ponto da exploração do espaço de soluções. A melhor solução encontrada nesse procedimento é então o ótimo global. A relaxação é parte tão fundamental nessa abordagem que geralmente é utilizada para classificar esses procedimentos [LAW85]. Como esses procedimentos podem levar ao ótimo global, costumam ser chamados de métodos exatos.

Os algoritmos destinados a encontrar soluções exatas para problemas

$\mathcal{NP}$-completos são construídos a partir de métodos capazes de determinar limites superiores e inferiores para a solução buscada e um esquema de enumeração. Para cada instância, os limites superiores e inferiores, ou seja, as soluções viáveis, vão sendo calculados, até que uma certa instância tenha que ser dividida em sub-problemas, de tal forma que a união desses sub-problemas conduza a soluções viáveis do problema principal. Os sub-problemas são processados de forma similar. Essas abordagens podem ser visualizadas como uma árvore onde cada nó corresponde a um problema e os filhos a sub-problemas. Esses algoritmos são classificados em duas grandes categorias: *Branch and Bound* e *Branch and Cut.*

Branch and Bound Nestes algoritmos os limites inferiores são derivados por meios puramente combinatórios. Relaxações discretas do TSP, por exemplo, são resolvidas empregando métodos discretos. A relaxação introduzida geralmente reduz o problema, tanto no caso do QAP como do TSP, a um problema do tipo *Linear Assignment Problem* (LAP), cuja solução se conhece em tempo polinomial[19] [RAR98]. Uma relaxação particularmente importante é a chamada *relaxação lagrangeana*, onde o problema de programação linear inteira tem algumas das restrições lineares relaxadas ao movê-las para a função objetivo com termos tais como:

$$\cdots + v_i \left(b_i - \sum_{j=1}^{n} a_{i,j} x_i \right) \cdots$$

onde v_i é um multiplicador de Lagrange. Caso as restrições a relaxar tenham a forma $\sum_{j=1}^{n} a_{i,j} x_i \geq b_i$ ($\sum_{j=1}^{n} a_{i,j} x_i \leq b_i$), o multiplicador deverá

[19] $O(n^3)$, por exemplo para o algoritmo Húngaro.

ser $v_i \geq 0$ ($v_i \leq 0$) para um modelo de minimização, enquanto as condições $\sum_{j=1}^{n} a_{i,j}x_i = b_i$ não impõem restrições aos multiplicadores v_i.

A relaxação lagrangeana teve mais sucesso na solução do TSP que no caso do QAP [LAW85] [MIY77] [ANS98a], sendo por isso mais freqüente nos algoritmos propostos para o TSP do que no QAP. Ainda assim, a técnica é relativamente mais eficiente para casos do TSP assimétrico que para o caso simétrico. No caso dos QAPs novas pesquisas têm-se voltado para *trust regions* [KAR93] [ANS98b] e programação semidefinida *semidefinite programming* [ZHA98b].

Estes algoritmos, entretanto, têm desempenho inferior às abordagens *Branch and Cut*.

Branch and Cut Nos algoritmos desta classe, os limites inferiores são obtidos por relaxação da programação linear. A primeira relaxação é a retirada da exigência de integralidade das variáveis, utilizando-se então a melhor aproximação inteira para decidir qual ramo da árvore de busca contém as soluções viáveis. A idéia central nesta classe de algoritmos é a de definir um plano de corte para limitar a necessidade de explicitar todas as inequações que formam o problema de programação linear, onde estas são adicionadas à medida que o problema vai sendo resolvido por técnicas de solução de programação linear clássicas, como por exemplo o algoritmo simplex [REI94].

3.4.2 Algoritmos Aproximados

Uma segunda maneira de abordar os problemas de otimização discreta é pela utilização de alguma heurística para encontrar soluções *próximas do ótimo* de forma rápida. Esses algoritmos são também denominados sub-ótimos. Quando se estuda esses tipos de algoritmos, a pergunta que surge imediatamente é: existe alguma forma de determinar *quão* distante da solução ótima está a solução corrente? Infelizmente, prova-se que para haver uma garantia que uma certa heurística encontra soluções com tamanho não maior que um certo fator, para *todas* as instâncias de um problema combinatório, seria necessário que $\mathcal{P} = \mathcal{NP}$ [SAH76], uma possibilidade que se mostra cada vez mais improvável. No caso do TSP que atende à desigualdade triangular, entretanto, já se tem resultados provando que é possível gerar soluções com no máximo 3/2 vezes o comprimento do percurso ótimo com um algoritmo com tempo de execução $O(n^3)$ [LAW85]. De forma geral, a avaliação de algoritmos heurísticos costuma incluir alguma prova de sua qualidade através de estatísticas, pois como afirmaram Lin e Kernighan [LIN73] "... para aplicações práticas, frequentemente o que importa é que boas respostas sejam obtidas em tempos de execução exeqüíveis."

Nos algoritmos aproximados, utiliza-se alguma heurística ou adaptação de um algoritmo que é insuficiente para resolver o problema do ponto de vista teórico para encontrar uma solução para o problema em questão. Por exemplo, no TSP parte-se de uma solução de determinação de uma *árvore mínima* utilizando o algoritmo de Kruskal ou Sollin [SAV80], de onde se obtém um primeiro percurso que passa por todas as cidades (supondo que se tenha de passar, na pior das hipóteses, por cada arco na ida e na volta) com

um comprimento de percurso de duas vezes o comprimento da árvore mínima. No QAP, uma heurística inicia reduzindo o problema a um LAP e para cada acréscimo de uma usina utiliza-se uma regra de impacto na solução viável, analisando a matriz de custo redefinida para as alocações faltantes usando as regras propostas por Gilmore [MIY77].

Para que os algoritmos aproximados tenham atrativo sobre os exatos, eles devem possuir a propriedade de ter tempos de execução mais curtos e oferecer soluções razoáveis para os problemas encontrados na prática. Em algumas abordagens, onde se desenvolvem sistemas para solução específica desses problemas de otimização, empregam-se geradores de soluções aproximadas para iniciar os algoritmos mais rigorosos com valores de limite superior da solução viável, auxiliando então a acelerar o tempo de execução destes últimos. Por exemplo, a solução obtida pelo algoritmo da árvore mínima dá um valor limite superior para as decisões de uma algoritmo *branch and bound*[20].

Soluções sub-ótimas por construção As soluções por construção tentam determinar uma solução viável gerando soluções parciais e sucessivamente aumentando o espaço do problema (por exemplo alocando usinas a localidades, no caso do QAP, ou adicionando cidades ao percurso, no caso do TSP) desde que não se violem as restrições do problema (por exemplo, visitar mais de uma vez uma cidade, no caso do TSP, ou alocar mais de uma usina à mesma localidade, no caso do QAP). As partes da solução já resolvidas não são mais modificadas até o término do algoritmo. Em outras palavras, não há retrocesso (*backtracking*) para rever um movimento do algoritmo n passos

[20]Note-se que neste caso não é nem necessário transformar a árvore mínima em um percurso, já que o dobro do seu comprimento é o limite superior buscado.

atrás.

Soluções por construção não tentam melhorar os valores obtidos ao término desses algoritmos, o que os torna determinísticos para cada instância do problema: a aplicação de um certo algoritmo P a uma instância I de um problema dará sempre o mesmo resultado $P(I)$ não importando quantas vezes se rode o algoritmo.

Não obstante o compromisso feito com o desempenho destes algoritmos quanto à otimalidade das soluções (lembremos que estes são os sub-ótimos), a complexidade temporal embora polinomial se situa entre $O(n^2)$ e $O(n^5)$, incluindo aqueles para TSP e QAP, sendo que para os primeiros temos ordens geralmente quadráticas e às vezes cúbicas, enquanto no QAP o típico situa-se nas ordens quadrática para cima [MIY77] [LAW85].

Soluções sub-ótimas por melhoria Nos algoritmos baseados em heurísticas de melhoria, inicia-se o processo como alguma solução gerada por outros meios (que no limite pode ter sido aleatoriamente) e procede-se a alterações segundo algum esquema que conduza a soluções de melhor qualidade (mais próximas do ótimo) do que a solução corrente do algoritmo. Assim, no caso do QAP pode-se procurar o par de usinas que tem o maior intercâmbio de mercadorias e trocar com outras duas, de modo que a distância entre elas seja a menor possível, ou no caso do TSP pode-se proceder a uma busca de cruzamentos dos arcos que ligam as cidades para trocar por arcos mais curtos.

Os algoritmos de melhoria costumam ser re-executados várias vezes para a mesma instância do problema uma vez que em função da solução inicial

pode-se chegar a soluções finais distintas[21]. Soluções finais são aquelas em que o algoritmo pára por não encontrar nenhum novo movimento que gere uma melhor solução viável. Ao fim de um certo número de execuções, a melhor solução final é escolhida. Em outras palavras, considerando como entrada desses algoritmos as soluções iniciais, eles são **sensíveis à instância** do problema.

3.4.3 Meta-Heurísticas

As heurísticas para solução de problemas de otimização discreta geralmente não conseguem encontrar a solução ótima global, mas sim algum *ótimo local*. A partir de um ótimo local, não se consegue gerar nenhuma melhora uma vez atingida aquela solução. Como exposto acima, uma forma de evitar que se chegue a uma solução muito aquém do mínimo global é a re-execução dos algoritmos várias vezes. Esse método mais óbvio no caso de soluções obtidas por melhoria pode também ser empregado no caso de algoritmos por construção. Por exemplo, no caso do TSP um dos algoritmos de construção mais simples é a chamada regra do vizinho mais próximo: iniciando-se numa cidade qualquer, localiza-se, na métrica do problema, a cidade mais próxima e adiciona-se ao percurso, buscando em seguida a próxima cidade não visitada, e assim sucessivamente. Pode-se chegar a percursos de tamanho diferentes iniciando por cidades diferentes. O sucesso dessas abordagens é, no entanto, limitado [REI94].

Uma outra possibilidade é a perturbação do fluxo de busca das soluções

[21] Por isso o uso da geração de soluções iniciais aletórias é atrativo em conjunto com estes.

que admita em passos intermediários soluções com valores acima do mínimo já encontrado até aquele momento, aumentando assim a chance de trilhar por caminhos no espaço de soluções que levem a mínimos menores e ultimadamente escapando de um mínimo local. O ingrediente fundamental nessas abordagens é a introdução de alguma aleatoriedade no procedimento.

Entre as abordagens nessa linha pode-se citar as motivadas por fenômenos termodinâmicos, nos algoritmos de recozimento simulado [KIR83], nos fenômenos de evolução nos algoritmos genéticos [DAV91] e no funcionamento do cérebro nas redes neuronais [POT93].

Nas abordagens que permitem soluções intermediárias em que não há uma melhora explícita na solução, há o risco do algoritmo entrar num ciclo (eventualmente infinito) se não houver algum esquema para eliminação de soluções repetidas. Uma forma eliminar este problema é o uso de listas que contêm movimentos temporariamente proibidos. Essas listas são chamadas de listas de tabus e algoritmos baseados nessa técnica são denominados de *tabu search* [GLO89] [GLO90].

Em geral, todas estas abordagens, denominadas meta-heurísticas [VIA98], são de aplicação mais geral, não se limitando a problemas de otimização discreta. São portanto atrativas para ambientes de resolução de problemas mais genéricos, pois muitas delas permitem tratar certos problemas com pouco conhecimento sobre sua estrutura e comparativamente a outras técnicas com pouco esforço. Entretanto, não está claro até agora se essas técnicas podem realmente competir com algoritmos mais especializados.

A título de exemplo de classificação, os algoritmos descritos nas seções 3.2 e 3.3 seriam classificados da seguinte maneira:

- o algoritmo Lin Kernighan é um algoritmo heurístico por melhoria;
- o algoritmo *greedy* é um algoritmo heurístico por construção;
- o algoritmo CRAFT é um algoritmo heurístico por melhoria;
- o algoritmo HGW é também um algoritmo heurístico por melhoria.

3.5 Formulação do Problema de Otimização de Inserção

Utilizando a notação desenvolvida na seção 2.6, particularmente a equação (2), apresentamos agora o objetivo da otimização deste trabalho bem como a métrica empregada. Mostramos também a dificuldade de solução, devido aos sub-problemas serem bem imbricados e também à explosão combinatória já que ambos os sub-problemas são $\mathcal{NP}$-difícil.

A otimização que desejamos efetuar é uma otimização no nível 3 (arranjo e seqüenciamento) conforme exposto na seção 2.4. Para fins de formulação, o problema de otimização da máquina insersora Panasert AVK será considerado com ela operando no modo conectado (seção 2.5), o que nos leva a um problema com um número máximo de 2000 componentes[22] a inserir e 120 escaninhos a serem utilizados[23].

A função objetivo a otimizar (minimizar) é o tempo total de inserção de uma determinada placa, matematicamente:

[22]Compatível com o tamanho total da memória da máquina, conforme discutido na seção 2.5.

[23]Considera-se que todos os componentes têm 26 mm de largura.

$$\min f(p,q) = \sum_{i=1}^{n} \tau(i) + c_2 \tag{21}$$

onde p é um percurso na placa, q uma atribuição de componentes aos escaninhos e $\tau(i)$ o tempo de ciclo da máquina para uma operação de inserção i dados p e q, conforme a equação (1).

Sejam as seguintes definições:

n: número total de inserções na placa ($n \leq 2000$);

m: número de tipos de componentes distintos a serem utilizados nas placas ($m \leq 120$);

x_{ij}: variável de decisão que representa o percurso da posição i para a posição j na placa de circuito impresso ($0 \leq i, j \leq n$);

y_{kl}: variável de decisão que representa a colocação do tipo de componente k no escaninho l;

$t(i)$: função que tem como entrada uma posição i na placa e retorna o tipo de componente que deve ser inserido nesta posição;

O problema é então minimizar a quantidade:

$$T = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} \sum_{l=1}^{m} c_{ijkl} x_{ij} y_{t_i k} y_{t_j l}, \tag{22}$$

sujeita a:

$$x_{ii} = 0, \qquad 1 \leq i \leq n \tag{23}$$

$$x_{ij} \in \{0,1\}, \qquad 1 \leq i,j \leq n \tag{24}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad 1 \leq j \leq n \tag{25}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad 1 \leq i \leq n \tag{26}$$

$$u_i - u_j + nx_{ij} \leq n-1, \qquad 2 \leq i,j \leq n \tag{27}$$

$$y_{kl} \in \{0,1\}, \qquad 1 \leq k,l \leq n \tag{28}$$

$$\sum_{k=1}^{m} y_{kl} = 1, \qquad 1 \leq l \leq m \tag{29}$$

$$\sum_{l=1}^{m} y_{kl} = 1, \qquad 1 \leq k \leq m \tag{30}$$

onde:

c_{ijkl} = é tempo necessário a uma inserção, igual ao tempo de duração de ciclo da máquina $\tau(i)$,

$x_{ij} = 1$ se e somente se existe movimento da posição i para a posição j,

$y_{kl} = 1$ se e somente se o componente no escaninho l é k.

u_i, u_j são números reais arbitrários, diferentes entre si.

A função objetivo (22) acumula o tempo total de inserção. As equações (25) a (27) são as restrições do TSP (seção 3.2), as equações (29) a (28) são restrições do QAP(seção 3.3).

Esta formulação é similar à encontrada por Leipälä e Nevalainen [LEI89] ao analisar uma máquina Panasert RH e àquela relatada relatada por Walas

e Askin [WAL84] ao estudar uma prensa puncionadora com um magazine rotativo de ferramentas. Nossa formulação difere da encontrada por Broad et al. [BRO96], pois a máquina estudada neste caso (Panasonic MPa) tinha escaninhos tanto na parte da frente como na parte de trás do porta placas (mesa XY). Assim, a cabeça tinha que efetuar um movimento mais complexo, necessitando levar em conta no equacionamento os arcos de retorno da cabeça a um dos dois conjuntos de escaninhos.

3.6 Combinação de Algoritmos

Como discutido na seção 3.4, os problemas de otimização discreta somente podem ser abordados por algoritmos heurísticos, exceto nos casos em que as instâncias dos problemas tenham tamanho bem reduzido, quando o uso de enumeração é exeqüível. Outrossim, percebe-se que caso se combinasse esses algoritmos de alguma forma, dever-se-ia obter soluções melhores do que as obtidas individualmente pela execução de cada um deles.

Em adição à seqüência óbvia de se ter um algoritmo de construção seguido por um de melhoria, não há nenhuma regra que nos permita determinar qual a forma de agrupá-los para obter o melhor resultado possível. Quando um sistema de uso geral é construído, como por exemplo o SAGATS [VIA98], faz-se uma série de compromissos e determina-se uma ordem de operação dos algoritmos. Para o TSP há propostas de implementação de pacotes de resolução desse problema, considerando três cenários, que vão de recursos de CPU escassos até a abundância para considerá-los virtualmente ilimitados [REI94]. Peixoto e Souza discutem essa questão, mostrando que quando a coleção de heurísticas à mão é grande, torna-se proibitiva a análise da

seqüência certa dos algoritmos para resolver um problema [PEI94a].

O que se necessita, então, é uma maneira de organizar os algoritmos para que operem com maior sinergia possível, permitindo que cooperem entre si, obtendo soluções que não seriam encontradas por cada um deles isoladamente [SOU93a].

Neste trabalho, optou-se por utilizar a técnica de A-Teams. Trata-se de uma nova técnica para resolução de problemas baseada na utilização simultânea e assíncrona de diversos agentes, formados por algoritmos heurísticos, que cooperam entre si. Tal técnica é descrita no capítulo seguinte.

4 TIMES ASSÍNCRONOS (A-Teams)

Neste capítulo, descrevemos a técnica empregada para a resolução do problema de otimização do processo de inserção automática de componentes eletrônicos.

4.1 Conceitos Básicos

Um time assíncrono (**A-Team**) é um arranjo de algoritmos (também denominados **agentes**) de forma que possam cooperar para atingir um objetivo desejado pelo seu projetista sem que haja necessidade de uma coordenação prévia entre esse algoritmos. A única comunicação entre esses algoritmos se dá através de soluções compartilhadas, armazenadas em repositórios denominados **memórias**. O fato de não requererem coordenação permite que se organizem A-Teams onde os tempos de processamento dos vários agentes sejam bem distintos, permitindo que se possa combinar algoritmos mais velozes que produzem soluções mais fracas com aqueles que, embora tenham como saída soluções de melhor qualidade, são normalmente mais lentos.

4.2 Arquitetura de A-Teams

A arquitetura de uma solução utilizando A-Teams é construída empregando-se apenas dois elementos básicos: agentes e memórias [TAL96], conforme a figura 6 abaixo.

- **Memórias** são coleções de soluções que podem ser lidas ou depositadas

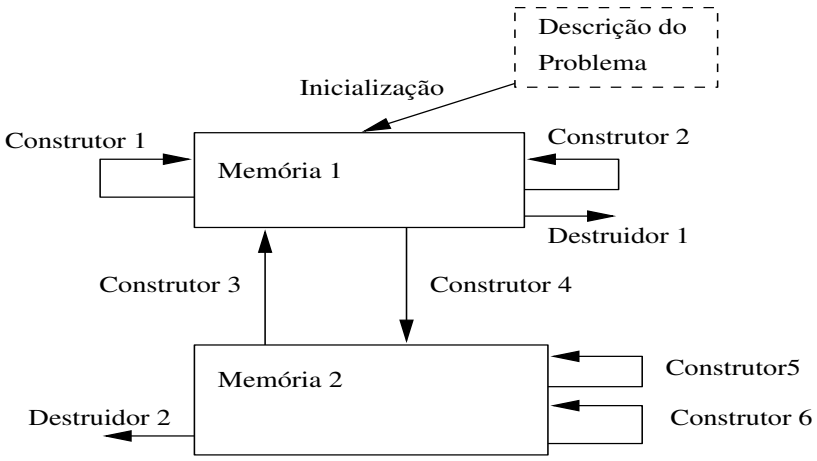

Figura 6: Arquitetura de um A-Team

(geradas) pelos agentes. Os agentes lêem as memórias para ter uma solução inicial a partir da qual podem gerar novas soluções, que por outro lado tornam-se disponíveis para os outros agentes que têm acesso a estas memórias.

- **Agentes** são unidades autônomas, cujo sistema de controle é totalmente autocontido. Não aceitam nenhuma instrução de seleção ou escalonamento de outros agentes ou qualquer outro tipo de elemento de controle externo. O agente tem três componentes: um operador (o algoritmo que gera a solução), um sistema de comunicação que estabelece como ele vai escrever e ler nas memórias compartilhadas, e um sistema de controle que determina quando e como vai trabalhar.

Os agentes podem ser de dois tipos: **construtores** e **destruidores**. Construtores são aqueles que colocam novas soluções (tentativas) nas memórias. Destruidores apagam soluções tentativas das memórias eliminando assim as menos apropriadas. Os destruidores podem ainda ser empregados para eli-

minar padrões de produção de soluções indesejadas tais como seqüências de soluções repetidas.

A-Teams podem ser vistos como uma rede de memórias e agentes. A propriedade fundamental dessa rede é o fato dos seus elementos serem **fortemente cíclicos**. Usando grafos para representar A-Teams [KAN97], temos os arcos representado os agentes e as memórias representadas pelos vértices.

A-Teams construídos dessa forma mostram ter propriedades identificadas nos algoritmos genéticos, redes neurais e técnicas de solução de problemas baseados no *tabu search* [GLO89] [GLO90], *blackboards* [NII96] e recozimento simulado (*simulated annealing*) [KIR83].

Soluções utilizando A-Teams já foram testadas em uma variedade de problemas de simulação [TAL96], e apresentam o seguinte comportamento em comum:

1. *Diversidade*: a qualidade das soluções melhora com a faixa de habilidades dos agentes construtores.

2. *Escala*: há pouca, se alguma, penalidade por excesso na habilidade dos construtores. Ao contrário, a escalabilidade parece ser o caso comum, já que a qualidade das soluções invariavelmente pode ser melhorada pela adição de agentes, tanto construtores como destruidores.

3. *Modularidade*: adicionar agentes autônomos a fluxos de dados fortemente cíclicos é relativamente fácil, não importando se os agentes são grandes ou pequenos, gerais ou especializados.

4. *Dualidade*: destruição apta pode compensar por construção inepta e

vice-versa.

5. *Tamanho da População*: a qualidade das soluções tende a melhorar com o aumento do tamanho das populações de soluções, embora haja uma tendência à saturação[24].

6. *Paralelismo*: a velocidade das soluções melhora conforme mais processadores (ou computadores) sejam adicionados até que haja processadores suficientes para todos os agentes trabalhar em paralelo todo o tempo. Geralmente, o aumento de velocidade é linear.

   Neste caso, acreditamos que esse comportamento se deva primordialmente ao fato dos tempos de processamento serem ordens de grandeza maiores que os tempos de comunicação, não sendo por isso observado os efeitos da lei de Ahmdahl [25]. Em particular, Peixoto e Souza [PEI94a] abordam essa a questão denominando essa característica dos A-Teams de granularidade grossa.

A técnica de A-Teams tem sido pesquisada desde 1990, quando foi feita a primeira aplicação com sucesso, e tem sido empregada na solução de problemas de otimização como o próprio TSP, projeto de arranha-céus, robôs

[24]Pesquisadores trabalhando com A-Teams para solução do TSP têm relatado bons resultados com memórias entre a metade e dobro do número de cidades [TAL96].

[25]A lei de Ahmdahl expressa a lei dos ganhos diminuentes em sistemas computacionais: a melhoria incremental na velocidade obtida pelo acréscimo de processadores diminui com o aumento de unidades. A razão se deve ao fato de que uma fração dos recursos adicionais não é utilizada completamente, em função do partilhamento de recursos, sincronização, etc. A título de exemplo, imagine-se um sistema com processadores os quais durante 50% do tempo ficam esperando por I/O; o aumento da capacidade de processamento seria 33% para dois processadores, 50% para quatro processadores, e 60%, 67% e 71% para 4, 5 e 6 processadores respectivamente[PAT90].

por demanda para o ônibus espacial, diagnóstico e controle de sistemas de energia elétrica e escalonamento de trens. Uma descrição mais completa com as devidas referências pode ser encontrada em [TAL96].

4.3 O Protocolo de Projeto de A-Teams

Ainda não se dispõe de uma teoria abrangente para orientação no processo de análise e síntese de A-Teams.

Os passos propostos a seguir derivam da experiência acumulada pelos experimentadores da técnica [TAL96] [PEI94b]:

[1] *Escolha do Problema*

Determinar em qual classe de problemas de otimização a solução buscada se encaixa.

[2] *Decomposição em sub-problemas*

A decomposição não precisa ser hieráquica. Os sub-problemas não precisam ser disjuntos ou distintos. Eles podem ter alguma sobreposição, ou alguma interação mais complexa. Os sub-problemas devem ser escolhidos de modo que possam ser conectados por agentes em laços fechados, onde cada sub-problema possa ser usado para construir soluções para o próximo sub-problema no laço.

[3] *Atribuição de uma ou mais memórias para cada sub-problema*

O propósito das memórias é guardar uma população das soluções tentativas de seu sub-problema. Populações maiores levam a melhores

soluções, mas os benefícios caem rapidamente. Populações de tamanho moderado têm desempenho esperado tão bom quanto as bem grandes.

[4] *Seleção dos algoritmos ou operadores para cada sub-problema*

Quanto maior a faixa das capacidades dos algoritmos, melhores serão as soluções encontradas. Não há necessidade de serem uniformes seja no tamanho ou na cobertura. Alguns podem ser grandes, outros pequenos, alguns gerais, outros especializados. O mais importante é a variadade de abordagens.

[5] *Transformação de cada algoritmo em um agente autônomo*

Um agente pode ser entendido tendo três componentes: um operador, um sistema de comunicação, e um sistema de controle.

[6] *Formação de destruidores*

Deve haver um balanço entre construção e destruição de soluções tentativas, caso contrário as memórias rapidamente ficarão saturadas. Destruidores têm mais duas funções: tornar as soluções que caem no seu espaço de saída quase inacessíveis aos construtores e filtrar padrões de construção indesejados, reconhecendo-os e eliminando-os.

[7] *Montagem dos agentes e memórias em fluxos de dados fortemente cíclicos*

[8] *Testes e modificação dos fluxos de dados*

O processo de resolução tem como primeiro passo a inicialização das memórias com populações iniciais de soluções tentativas, ativação dos agentes e monitoramento das mudanças nas populações de soluções. Se as soluções convergem lentamente, deve-se retornar ao passo [4]. Se mesmo assim a convergência for lenta, deve-se reiniciar o passo [2].

4.4 Adequação da Técnica ao Problema de Otimização de Inserção

A adequação da técnica de A-Teams ao problema da inserção automática de componentes eletrônicos fica agora nítida. Trata-se de um problema de otimização complexo, que mesmo quando dividido em sub-problemas somente pode ser solucionado por algoritmos heurísticos, que são sabidamente sensíveis à instância do problema. No caso de A-Teams, pode-se combinar um número de algoritmos para tentar cobrir uma faixa maior de casos. Ela é ainda atrativa pelo fato de ter simplicidade na formulação, permitindo que uma vez que se tenha escolhido uma ou mais representações[26] para o problema ou sub-problemas, e eleito os algoritmos mais promissores, já se possa trabalhar nas soluções.

Do ponto de vista da implementação, a técnica permite a reutilização de algoritmos já desenvolvidos e testados por outros pesquisadores. É evidente que deve-se proceder à etapa de transformação destes algoritmos em agentes, conforme será visto no próximo capítulo. Tal esforço, todavia, é muito inferior àquele que seria dispendido para criar os algoritmos desde o início. Um outro efeito positivo é o de permitir uma heterogeneidade em termos de linguagens de programação utilizadas no desenvolvimento dos agentes. Um último ponto a ressaltar é a extrema modularidade da técnica: na ausência de uma solução aceitável, basta tentar adicionar novos agentes, e com esforço reduzido, verificar se obtém-se uma melhor solução.

[26] Tipos de dados e forma de agregá-los.

5 O SISTEMA OPTIMA

Neste capítulo, descrevemos o sistema OPTIMA, detalhando a arquitetura de A-Teams empregada para solucionar o problema proposto nesta dissertação. Mostra-se, em particular, a escolha dos algoritmos, a forma de representação dos dados na(s) memória(s), e a execução do protocolo de projeto.

5.1 Descrição Geral

O sistema OPTIMA, como mostra a figura 7, é composto de um A-Team projetado para resolver o problema geral de otimização de inserção de componentes da AVK, considerando apenas o caso do uso dos escaninhos para conter um único tipo de componente por escaninho (conforme a célula "TSP + QAP", representada em negrito, da tabela 1). Em outras palavras, este sistema não endereça o problema adicional do problema de recuperação de componentes citado na seção 3.1.

5.2 Aplicação do Protocolo de Projeto

A seguir, detalhamos cada passo do protocolo de projeto utilizado na concepção do sistema, conforme a descrição da seção 4.3.

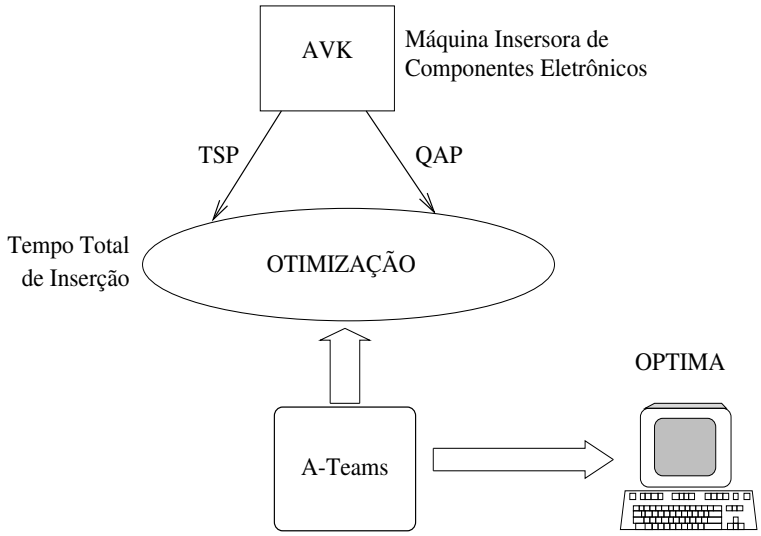

Figura 7: Visão Geral do Sistema OPTIMA

5.2.1 Escolha do Problema

O problema escolhido para ser resolvido com a técnica de A-Teams é a otimização do processo de inserção automática de componentes eletrônicos usando a máquina insersora AVK descrita na seção 2.5. A forma de exploração da máquina é o modo *conectado*, e a otimização visa otimizar o arranjo de escaninhos e a ordem de inserção considerando para cada problema uma placa de circuito impresso de cada vez.

As implicações dessa especificação são as seguintes:

- conforme descrito no apêndice A, a sintaxe do formato dos programas de inserção permite instruções de salto. A existência de tais instruções justifica-se na prática pois na falta de determinado tipo de componente ou falha na furação da placa de circuito impresso pode-se ainda

proceder à montagem da placa, mesmo que incompleta. No sistema OPTIMA, é realizado um pré-processamento do arquivo de entrada de modo a eliminar tais instruções.

- o problema é pré-processado também para deixar todos os escaninhos adjacentes, respeitando as restrições da largura das fitas de componentes, conforme descrito na seção 2. Dessa forma, condiciona-se o QAP, pois na formulação adotada nos algoritmos utilizados no OPTIMA, o número de plantas deve bater com o número de localidades.

5.2.2 Decomposição em sub-problemas

O problema de otimização, descrito no capítulo 3, pode ser decomposto em dois sub-problemas o TSP e o QAP, como discutido na seção 3.1.

Ressalte-se aqui a adequação da técnica de times assíncronos a este problema. Como já elencado na seção 4.3, o projeto de A-Teams não exige que a decomposição dos sub-problemas formem problemas disjuntos ou hieráquicos. A otimização de máquinas insersoras do tipo da AVK recai em uma mescla do TSP+QAP dos quais não há uma hierarquia nem maneira não arbitrária de separá-los, exceto em casos muito específicos discutidos na seção 3.1 e sumarizados na tabela 1.

5.2.3 Atribuição de uma ou mais memórias para cada sub-problema

Na construção do OPTIMA, optamos por utilizar uma única memória para todos os agentes, uma vez que a forma de representar as soluções são arquivos

no formato da insersora conforme descrito no Apêndice A.

Assim, tanto os agentes que resolvem os casos de QAP como TSP leêm como entrada um arquivo no formato do programa da AVK e retornam como saída um novo arquivo para a memória também nesse formato. Este arquivo poderia ser até eventualmente executado na insersora.

Essa abordagem simplifica a implementação do sistema como um todo, e em particular a integração de novos agentes ao sistema.

5.2.4 Seleção dos algoritmos ou operadores para cada sub-problema

A seleção dos algoritmos utilizados nos agentes teve como critério suas propriedades teóricas (complexidade, velocidade, etc.), e a disponibilidade de código fonte, com implementações já testadas. Alguns destes códigos encontram-se publicados em papel [PRE92] e outros disponibilizados na Internet como a biblioteca TOMS `http://netlib.bell-labs.com/netlib/toms`[27] ou `http://www.ing.unlp.edu.ar/cetad/mos/`.

Os algoritmos exatos foram imediatamente descartados para o caso do QAP, uma vez que o tamanho máximo do problema na AVK (120) é bem maior do que o máximo atualmente conseguido com essa abordagem [HAH98] [CUN95].

Já no caso do TSP, em princípio parecia possível lançar mão de alguma implementação, dado o tamanho típico das placas estar em torno de 200 com-

[27] Parte desse acervo encontra-se também disponível em CD-ROM da Walnut Creek "netlib", June 1997.

ponentes. Contudo, também para o TSP os algoritmos baseados em *branch and bound* mostraram ainda não ser capazes de resolver o problema com recursos computacionais típicos em instalações industriais [CUN95] [CAR95]. Assim, descartou-se sua utilização no OPTIMA.

Como a técnica dos A-Teams pressupõe uma inicialização nas memórias com soluções iniciais geradas aleatoriamente, os algoritmos utilizados na formação dos construtores dos A-Teams costumam ser preferencialmente algoritmos aproximados de melhoria (seção 3.4), para que essas soluções iniciais possam ir sendo sucessivamente modificadas, de modo a gerar soluções cada vez mais viáveis. Entretanto, no caso do OPTIMA o uso de algoritmos de construção também se mostra interessante uma vez que cada nova solução do QAP (TSP) condiciona uma nova instância do TSP (QAP), se considerássemos apenas a solução desses problemas isoladamente.

Algoritmos para Resolver o Problema do Caixeiro Viajante O tipo de TSP que se formula ao resolver o problema de otimização da inserção automática na AVK é um TSP simétrico e que satisfaz a inequalidade triangular.

Estamos utilizando no OPTIMA uma implementação do algoritmo Lin-Kernighan, empregando movimentos 3-opt como elemento básico e número limitado em avanço de submovimentos [REI94], e uma implementação do algoritmo *greedy* [RAR98], ambos discutidos na seção 3.2.

Algoritmos para Resolver o Problema da Atribuição Quadrática O QAP formulado para analisar o problema de otimização da inserção au-

tomática na AVK é assimétrico, já que a matriz de fluxo representa o número de trocas (saltos) entre escaninhos, e a troca de um escaninho para outro é condicionada pela direção do percurso de inserção na placa. Assim, é possível haver fluxo somente de um escaninho para um consecutivo, por exemplo. É também do tipo euclidiano, uma vez que a matriz de distâncias é simétrica e a desigualdade triangular também se verifica.

Os algoritmos utilizados nesta implementação foram o CRAFT [ARM63] e o HGW [WES83], ambos algoritmos heurísticos por melhoria, discutidos na seção 3.3.

Determinação de Limites para os Algoritmos Na solução de TSPs ou QAPs puros, a determinação dos limites (*bounds*), sejam superiores ou inferiores, é necessária para que certos algoritmos possam tanto executar mudanças na sua operação, bem como se possa verificar o andamento da resolução do problema [VAL97] [MAN95] [HAD98a] [HAD98b] [RAM98].

Tais limites são também utilizados para determinar quão afastada da possível solução ótima se encontra uma dada solução corrente gerada por um determinado algoritmo [LAW85] [BUR96].

No caso da AVK, pode-se usar como limite inferior (*lower bound*) do tempo total de inserção o valor do número de componentes a inserir vezes o inverso da cadência da máquina. Com essa métrica auxiliar, pode-se calcular a qualidade das soluções obtidas por qualquer combinação de algoritmos, determinando quão maior são seus tempos em relação a esse tempo mínimo teórico.

5.2.5 Transformação de cada algoritmo em um agente autônomo

Os algoritmos escolhidos resolvem instâncias canônicas dos sub-problemas TSP e QAP. Para fazer uso deles, é necessário adaptar cada algoritmo para que possa receber dados (da instância) do problema da AVK. Esse procedimento será denominado de *extração dos sub-problemas*.

Os algoritmos devem então estar munidos de um escalonador (*scheduler*) que busque instâncias do problema para processar, ative-os e execute critérios de parada. Tal aspecto encontra-se mais detalhado na seção 5.3.

Extração dos sub-problemas Para a obtenção dos sub-problemas TSP e QAP temos que processar a definição do problema, dada inicialmente no formato de arquivo de programa de inserção descrito no Apêndice A, obtendo-se assim a matriz de distância para o TSP e as matrizes de distância e fluxo para o QAP.

TSP A matriz de distâncias para o TSP é obtida empregando-se a equação 1, e considerando os $n(n-1)/2$ custos para os possíveis pares de inserção, dado que a operação da máquina faz com ela tenha custos (tempos) simétricos.

Nos algoritmos TSP de melhoria que necessitam de um percurso inicial para iniciar o processamento, a ordem do arquivo de programa de inserção é naturalmente um percurso possível do TSP[28].

[28]Num grande número de implementações de algoritmos TSP, a ordem dada pela matriz de distâncias (ou seja o percurso trivial $1, 2, 3, \ldots, n$) é considerada a rota inicial.

QAP No caso do QAP temos de considerar as matrizes de distâncias e de fluxo. A matriz de distâncias sempre será uma tabela com diagonal nula indicando o tempo de trocas entre escaninhos, sendo monotonicamente crescente para escaninhos mais distantes entre si:

$$\begin{array}{ccccc} 0 & t_1 & t_2 & \cdots & t_n \\ t_1 & 0 & t_1 & \cdots & t_{n-1} \\ \vdots & & \ddots & & \\ t_{n-1} & & & \ddots & \\ t_n & t_{n-1} & \cdots & \cdots & 0 \end{array}$$

Essa matriz é geralmente fixa para a máquina, exceto se os operadores efetuarem ajustes na insersora, que modifiquem suas velocidades ou qualquer outro parâmetro que influa no tempo de troca de escaninhos.

Na prática, para o caso concreto da AVK, as entradas nessa matriz para valores com troca de mais de cinco escaninhos de distância têm um valor artificialmente alto, para forçar os otimizadores a não os escolherem. Tal decisão se justifica pelo fato de alguns operadores da máquina terem informado que o conjunto de escaninhos se perde caso se programe um salto de escaninhos com tal distância. Dito de outra forma, tal decisão garante a geração de um programa correto, conforme descrito na seção 2.4.

A matriz de fluxo será uma função do caminho que a máquina insersora vai percorrer na placa, dando como resultado quantas vezes ocorrem trocas entre dois determinados escaninhos. Essa matriz é uma matriz de banda (esparsa), pois devido à exigência de exploração da máquina que recomenda que não se efetuem saltos de escaninhos de mais de 5 posições, não haverá

também fluxo para escaninhos com esse afastamento. A diagonal pode ser não nula, quando as inserções sucessivas ocorrem com componentes do mesmo tipo, usando-se o mesmo escaninho. Como esses casos não contribuem com custo para a solução do QAP, zera-se essas posições na extração das matrizes, tornando-as adequadas para os algoritmos de QAP.

A obtenção da matriz de fluxo, a partir do programa de inserção que define o caminho, é feita da seguinte maneira:

Para cada nova inserção a ser executada, determina-se o escaninho corrente Ec e o novo escaninho Ec+1 e incrementa-se a contagem da matriz M[Ec][Ec+1]. O script map2qap.awk apresentado no Apêndice B executa a extração das matrizes do QAP a partir de um arquivo no formato de programa de inserção da AVK.

5.2.6 Formação de destruidores

Na técnica de A-Teams, pode-se ter destruição inepta caso se tenha construção apta. No caso, o destruidor vai apagando da memória n soluções piores cada vez que a população da memória exceda o tamanho máximo estipulado da população. Com esse critério, cria-se uma histerese que confere uma certa estabilidade ao A-Team.

5.2.7 Montagem dos agentes e memórias em fluxos de dados fortemente cíclicos

A arquitetura do A-Team tem como ponto forte o fluxo de dados fortemente cíclico. No nosso caso, onde todos os agente operam numa mesma memória, estamos assegurando esta característica.

5.3 Arquitetura do Sistema OPTIMA

A implementação do OPTIMA ficou muito flexível, permitindo que facilmente se incorporem mais agentes, necessitando-se apenas que seja feita uma pequena adaptação, como descrito a seguir. Note-se que devido aos agentes serem programas autônomos, o OPTIMA permite que se tenha agentes programados em linguagens diferentes, simplificando ainda mais a integração no sistema.

5.3.1 Agentes

Para serem usados em um A-Team, os algoritmos precisam ser transformados em agentes autônomos[KAN97].

Essa transformação é efetuada incluindo um *escalonador* para que cada agente decida quando e quão freqüentemente irá rodar e um *seletor* para escolher as soluções da(s) memória(s) que irá modificar (ou eventualmente destruir se for um agente destruidor).

O seletor escolhe uma solução utilizando os horários de criação (*time*

stamps) dos arquivos e compara com uma variável interna para evitar a repetição do uso das soluções.

Os agentes foram adaptados através de módulos. Alguns destes módulos transformam a representação do problema em formato do arquivo de programa da AVK nas estruturas de dados apropriadas para cada algoritmo, como as tabelas de distância para o TSP e nas tabelas de distância e fluxo para o QAP. Outros transformam a saída do algoritmo novamente no formato do arquivo de programa da máquina insersora. Desta forma, normaliza-se a representação do problema para cada algoritmo. Tal arquitetura dos agentes está representada na figura 8.

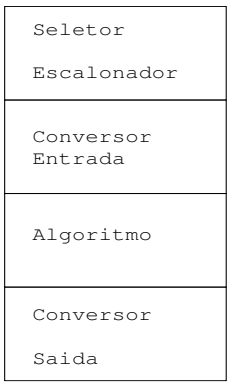

Figura 8: Arquitetura de um Agente

O destruidor monta uma tabela das soluções existentes no diretório e as ordena (inversamente) pelo tempo total calculado usando o modelo da máquina AVK. Quando o número de soluções excede um valor pré-determinado, determinado pelo usuário através da IHM, ele apaga as soluções que tenham os maiores tempos totais de inserção, eliminando assim as menos promissoras e abrindo espaço na memória do A-Team.

5.3.2 Memória

No OPTIMA a memória é um diretório cujo nome passa a ser o da corrida de otimização, e onde as soluções são armazenadas em arquivos.

Devido a não ser necessário utilizar algoritmos que necessitem de representações parciais (como por exemplo o deconstrutor proposto por [SOU93] ou o Held-Karp [HEL70] para o TSP), que necessitariam de diferentes memórias para armazená-las, a arquitetura do A-Team para o OPTIMA permanece com uma única memória e os agentes modificando as soluções ali depositadas, conforme mostra a figura 9.

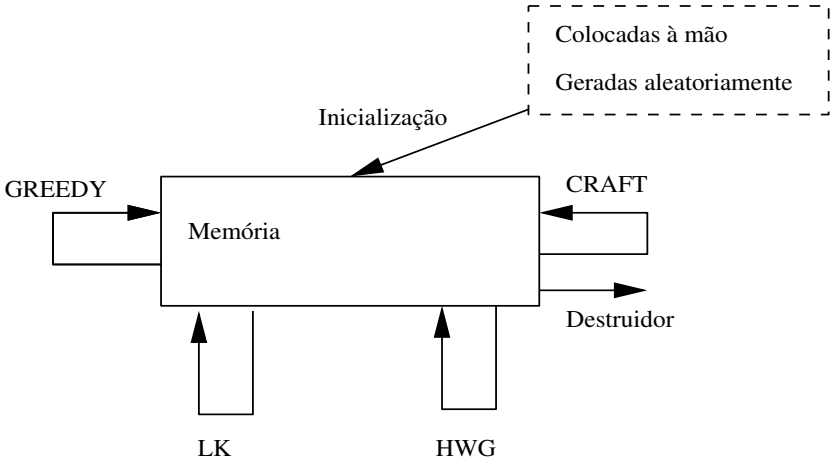

Figura 9: Arquitetura do Sistema OPTIMA

5.3.3 Controle

O usuário pode iniciar uma corrida de otimização no OPTIMA usando dois critérios de inicialização: semeadura de soluções iniciais e reutilização de soluções de uma outra corrida.

Quando é escolhida a semeadura de soluções na inicialização do A-Team, soluções iniciais são geradas aleatoriamente designando atribuições de tipos de componentes aos escaninhos e seqüências de inserção, de modo a criar uma população mínima para os agentes iniciarem seu trabalho. O sistema permite que se escolha o número inicial dessas soluções iniciais.

Na reutilização, inicia-se o sistema com os resultados de uma otimização anterior, sem modificar os arquivos existentes na memória.

Quanto ao critério de parada, podem ser utilizadas três estratégias distintas:

- limitar o tempo total de processamento do OPTIMA nos computadores escolhidos para a corrida;
- estabelecer o tempo total de inserção na insersora AVK (do problema), sendo que o OPTIMA para ao encontrar a primeira solução com tempo menor ou igual ao lançado nesse parâmetro;
- limitar o número de ciclos dos agentes, de modo a parar seu processamento quando esse limite é atingido[29].

[29]Caso algum agente não consiga encontrar nenhuma solução após examinar todas as soluções tentativas na memória do A-Team, ele pode parar antes de atingir esse limite.

5.4 Interface com o Usuário

Na implementação do OPTIMA, os agentes são programas que são lançados em função das escolhas efetuadas na Interface-Homem Máquina[30], e as memórias são pura e simplesmente diretórios, onde as soluções são armazenadas em arquivos. Embora a arquitetura do OPTIMA use uma única memória, para cada corrida cria-se um diretório com nome diferente, de modo a possibilitar a análise e resultados de parametrizações diversas.

No subdiretório onde está instalado o OPTIMA, existe um arquivo de configuração (optima.ini) que contém as informações dos agentes disponíveis[31].

O OPTIMA tem sua operação iniciada dando-se o comando `optima` no diretório de instalação do sistema, que inicia a IHM.

No painel "Opções" (figura 10) parametriza-se a otimização. No seletor de problemas, escolhe-se o arquivo no formato AVK (extensão .NCD) que se deseja otimizar. Em "Nome da Corrida", escolhe-se um nome para a corrida de otimização, que será usado para denominar o subdiretório a ser usado como memória do A-Team. Ainda neste painel, seleciona-se o número médio de soluções que se vai manter na memória durante a operação do A-Team, assim como o critério de parada, conforme definido na seção 5.3.3.

No painel "Agentes" (figura 11) selecionam-se os algoritmos que se de-

[30] Por concisão, doravante nos referiremos à Interface Homem-Máquina pela abreviação IHM.

[31] Para o sistema funcionar em mais de um computador, deveria-se fornecer também uma lista dos hosts. Além disso, o host onde será lançado o OPTIMA deveria ter seu diretório acessível e com permissão de escrita para os outros hosts que participarão no A-Team.

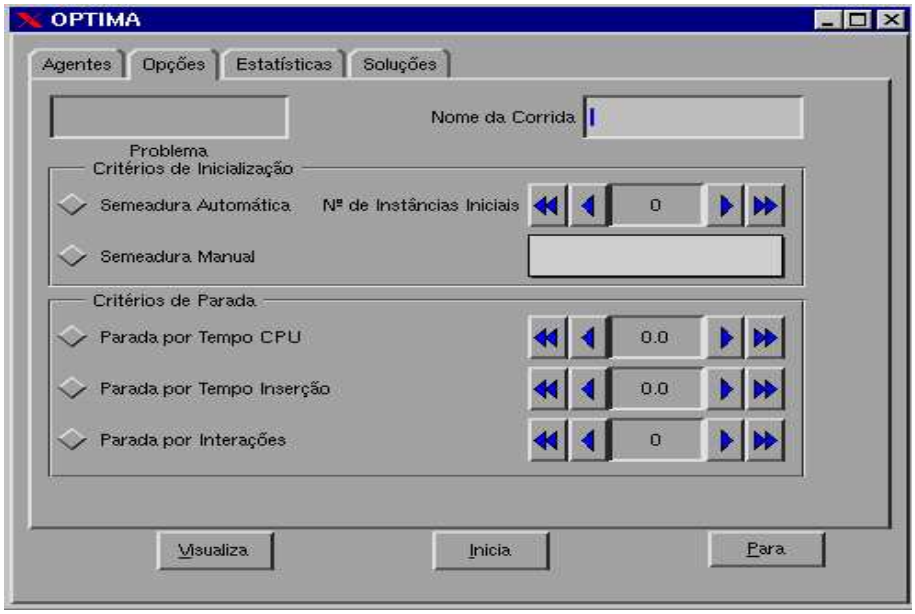

Figura 10: IHM do OPTIMA. Painel de opções

seja utilizar na corrida de otimização. Quando o OPTIMA é instalado para processamento distribuido em mais de um computador [32], pode-se selecionar em qual computador o agente será executado.

O botão "Visualiza" dispara um visualizador (figura 12) que permite verificar o percurso de inserção na placa de circuito impresso descrita pelo arquivo de entrada do problema para as várias soluções depositadas pelos agentes na memória do A-Team do OPTIMA. Pode-se ainda verificar o tempo de inserção, o número de trocas de escaninhos, e as distâncias percorridas nas direções X e Y.

O painel "Soluções"permite ao usuário visualizar uma lista disponível de soluções conforme mostra a figura 13. Permite também escolher a solução

[32]Esta opção não se encontra disponível no momento.

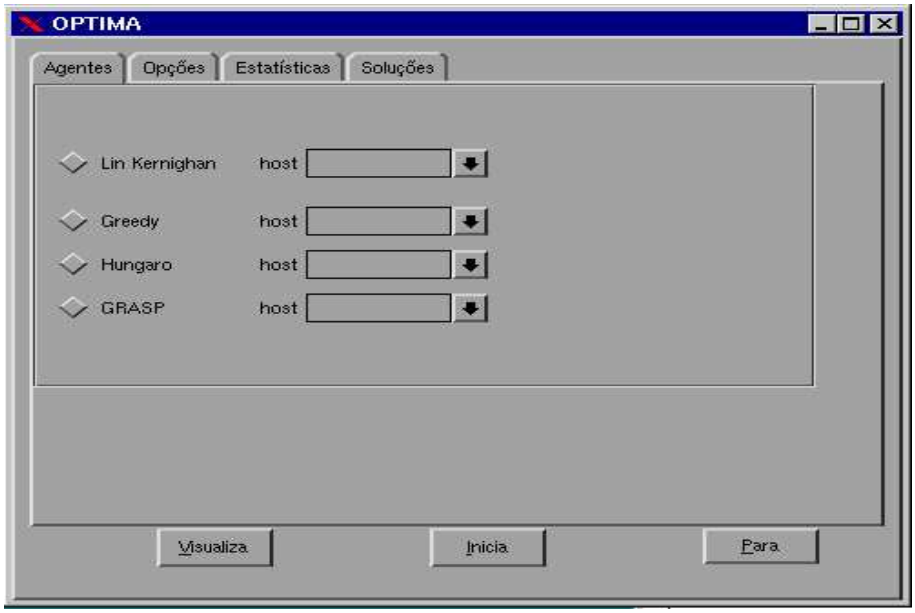

Figura 11: IHM do OPTIMA. Painel dos agentes.

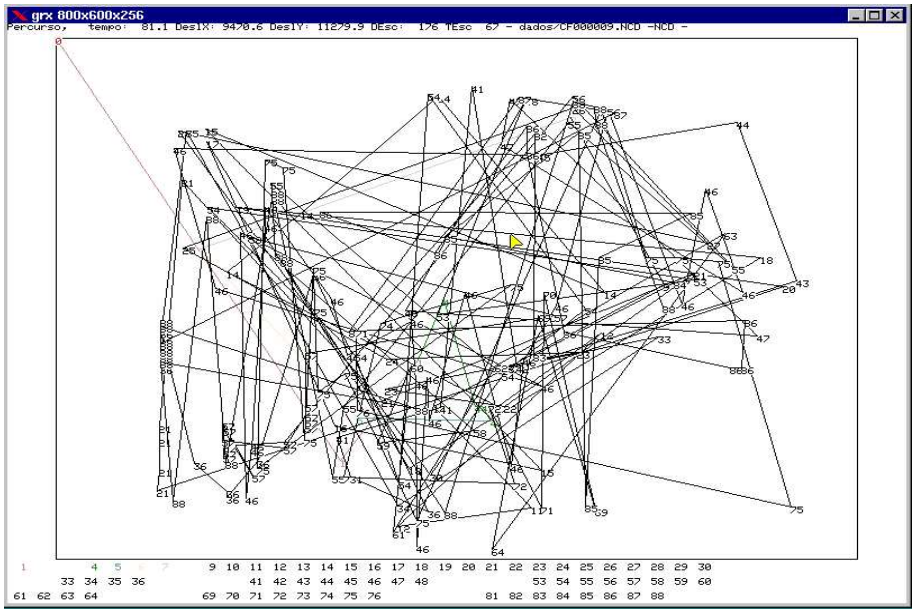

Figura 12: IHM do OPTIMA. Visualizador.

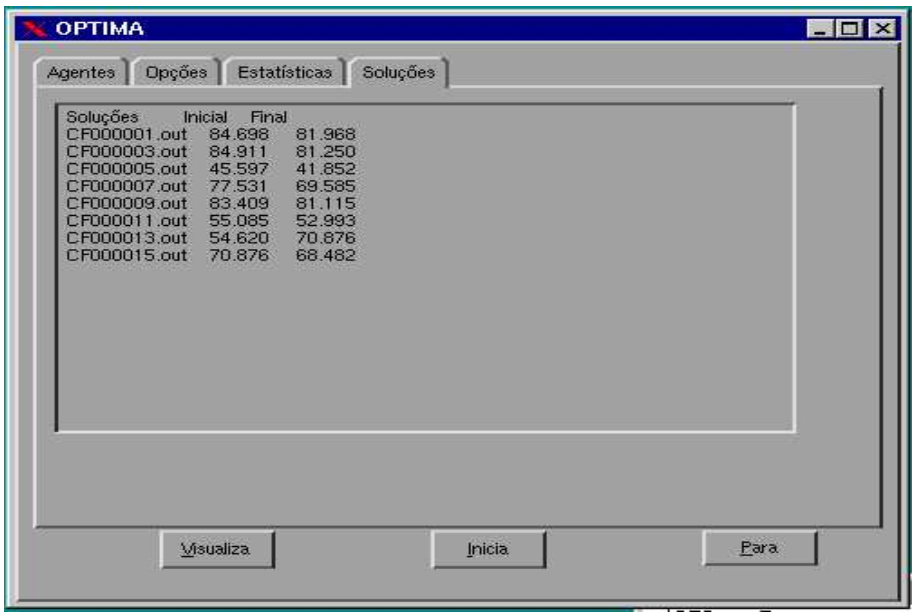

Figura 13: IHM do OPTIMA. Painel de informação sobre soluções.

entre as primeiras listadas, já que se encontram ordenadas de forma crescente por tempo total de inserção.

O início da otimização se dá clicando o botão "Inicia". Os agentes irão então selecionar instâncias do problema quer semeadas por embaralhamento do problema original ou a partir de soluções já encontradas em outras corridas, gerando eventualmente soluções melhores que serão depositadas na memória. Em ambos os casos, cabe ao usuário determinar estas soluções iniciais.

Além das soluções, o sistema gera também para cada corrida um arquivo de log (<nome_agente>.log) para cada agente, onde as informações de processamento são armazenadas. Estas podem ser examinadas através de um editor de texto.

O painel "Estatísticas" destina-se a uma expansão futura do sistema, descrita no capítulo 7.

6 AVALIAÇÃO e RESULTADOS

Neste capítulo, apresentamos os testes realizados no sistema OPTIMA, assim como uma análise dos resultados obtidos. Os testes referem-se tanto a problemas reais, isto é, seqüências de inserção de placas de circuito impresso de produtos existentes, como a problemas gerados sinteticamente, cujo objetivo é o de analisar alguns aspectos de operação do otimizador.

Realizamos a otimização de oito problemas reais, cedidos por um fabricante que produz placas de circuito impresso em regime de OEM, e de dois problemas criados sinteticamente, descritos a seguir.

6.1 Análise de Problemas Reais

Os oito problemas reais que foram testados pelo sistema OPTIMA têm as seguintes características:

CF000001 placa com quatro repetições do mesmo padrão de inserção, com uma mistura de percursos curtos e longos da mesa XY, e como conseqüência dessas repetições todos os tipos de componentes são obrigatoriamente inseridos mais de uma vez.

CF000003 placa com distribuição esparsa de componentes, com inserções únicas de certos tipos de componente, e repetições de componentes tanto espalhadas por toda a área da placa como localizadas numa região mais definida.

CF000005 placa com ocupação predominante de um único quadrante, havendo apenas três inserções que se localizam longe dessa concentração maior de componentes, contendo também uma mescla entre inserções únicas de certos tipos de componente até o caso de um tipo particular que é inserido onze vezes.

CF000007 placa com distribuição esparsa de componentes, com uma leve concentração num dos lados, e concentrações de repetições de tipos de componentes também espalhados pela área da placa.

CF000011 placa com quatro repetições, sendo cada padrão mais concentrado que no caso da CF000001, fazendo que haja uma região vazia em forma de cruz na placa onde não há nenhuma inserção, com número pequeno de tipos de componentes.

CF000013 placa sem repetição do padrão, porém também com três regiões mais definidas de inserção, um quadrante com apenas duas inserções, e variabilidade de repetições de tipos de componentes desde casos com uma única vez até treze repetições.

CF000015 placa com duas repetições do mesmo padrão, com pequeno espaço entre essas duas áreas de maior concentração, com repetições de tipos de componentes, em geral de três ou quatro.

CF000019 placa sem repetições com uma simetria ao longo de uma diagonal, sem espaço livre nesse eixo de simetria.

A configuração de tais problemas está descrita na tabela 2. Na coluna classe, utilizam-se as classes de problemas definidas na tabela 1.

Placa	Componentes	Escaninhos	Classe
CF000001	237	21	TSP + QAP
CF000003	194	66	TSP + QAP
CF000005	113	60	TSP + QAP
CF000007	183	93	TSP + QAP
CF000011	149	14	TSP + QAP
CF000013	120	48	TSP + QAP
CF000015	185	32	TSP + QAP
CF000019	104	37	TSP + QAP

Tabela 2: Casos de Teste Reais

Como visto anteriormente, o número de componentes indica o tamanho do TSP, e o de escaninhos o tamanho do QAP a serem resolvidos para otimizar a inserção nessas placas.

Na tabela 3, são exibidos os resultados para estes casos de teste. Nas colunas desta tabela, indicam-se respectivamente o tempo inicial com a configuração do fabricante, o tempo de inserção obtido com o OPTIMA, utilizando apenas os agentes para resolver o TSP, a redução percentual correspondente, o tempo de inserção obtido com a adição dos algoritmos para resolver o QAP, sua contribuição para a redução percentual correspondente, e finalmente a redução de tempo percentual total em relação ao tempo de inserção do fabricante.

Uma primeira análise desses dados mostra que o tempo de inserção obtido com o OPTIMA foi inferior ao fornecido pelo fabricante em todos os casos. Além disso, observou-se que ao tratar algumas instâncias o OPTIMA foi mais bem sucedido do que ao tratar outras, sendo que a redução média de tempo

Placa	Fabricante [s]	TSP [s]	% TSP	TSP + QAP [s]	% QAP	% Total
CF000001	81.97	80.19	2.17	80.19	0.00	2.17
CF000003	81.25	72.53	10.74	72.30	0.31	11.01
CF000005	45.60	42.22	7.40	42.09	0.31	7.69
CF000007	77.53	69.96	9.76	69.50	0.66	10.36
CF000011	55.09	52.35	4.97	52.29	0.10	5.07
CF000013	53.93	46.31	14.12	46.28	0.08	14.18
CF000015	68.48	65.05	5.01	64.96	0.13	5.14
CF000019	42.72	38.70	9.41	38.46	0.61	9.96

Tabela 3: Resultados Iniciais Obtidos para os Casos de Teste Reais

de inserção para todas as placas foi da ordem de 7,8%.

Contudo, devemos ressaltar que essas placas, por estarem em produção já há bastante tempo, são também produto de um procedimento de otimização manual, envolvendo várias horas de trabalho de técnicos. em particular, as placas com tempos de inserção mais otimizados correspondem àquelas de produção mais freqüente pelo fabricante, com conseqüente aumento de sintonia no programa de inserção.

Para esclarecer esse ponto de vista, montamos a tabela 4 onde exibem-se o tempo inicial correspondente ao tempo médio obtido pelo embaralhamento aleatório das permutações das linhas do programa e da atribuição dos escaninhos, o tempo obtido pelo sistema, e a redução percentual do tempo de inserção. Esse tempo inicial pode ser encarado como o tempo de inserção que seria obtido por um operador procurando fazer o primeiro programa correto para inserção da placa, usando, por exemplo, a ordem de aparecimento dos componentes em uma lista de material e lançando as coordenadas dos com-

Placa	Inicial [s]	OPTIMA [s]	% Total
CF000001	123.06	80.19	34.8
CF000003	107.49	72.30	32.7
CF000005	61.52	42.09	31.6
CF000007	102.16	69.50	32.0
CF000011	74.89	52.29	30.2
CF000013	68.16	46.28	32.1
CF000015	98.54	64.96	34.1
CF000019	55.76	38.46	31.0

Tabela 4: Resultados Intermediários para os Casos de Teste Reais

ponentes, valores de largura de dobra, etc. Comprova-se nesta tabela que as reduções percentuais obtidas com o sistema são respeitáveis, habilitando-o a produzir soluções de qualidade sem a necessidade da concentração de esforços de especialistas com recursos disponíveis na indústria.

Outra questão que a tabela 3 ressalta é a aparente pouca contribuição dos algoritmos de QAP para a melhoria da solução. Numa primeira análise, fomos levados a pensar que a qualidade das soluções obtidas dos algoritmos utilizados não fosse suficiente, chegando a considerar a hipótese de lançar mão de mais algoritmos. Contudo, a análise da operação individual dos mesmos indicava que quando calculavam os custos pelo produto das matrizes de fluxo e distância achavam melhorias bem mais altas, sendo que em alguns casos estas ultrapassavam os 20%.

A análise dessa questão indica que quando se extrai essas matrizes da forma exposta no capítulo anterior, não se considera a informação referente à oportunidade de aproveitar o tempo devido às operações concorrentes, con-

forme indicado na figura 3. Para tentar elucidar essa questão, modificou-se o algoritmo CRAFT para que ele calculasse o impacto das permutações usando a equação (1) do tempo de duração de ciclo da insersora ao invés de utilizar o custo calculado pelas matrizes de fluxo e de distância. Os resultados desses testes encontram-se na tabela 5. Vê-se que a utilização do algoritmo adaptado dessa forma consegue melhorar o tempo de inserção em 11,1%, em média.

Placa	Fabricante [s]	OPTIMA [s]	% Total
CF000001	81.97	77.55	5.4
CF000003	81.25	69.12	14.9
CF000005	45.60	40.62	10.9
CF000007	77.53	66.18	14.6
CF000011	55.09	49.68	9.8
CF000013	53.93	44.50	17.5
CF000015	68.48	64.46	5.9
CF000019	42.72	37.88	11.33

Tabela 5: Resultados Finais para os Casos de Teste Reais

6.2 Análise de Problemas Sintéticos

Estes problemas foram gerados de forma a se verificar o resultado por mínima inspeção, ou seja, possibilitando que a qualidade das saídas otimizadas pudesse ser verificada através da simples verificação dos resultados no visualizador.

Os problemas sintéticos testados foram os seguintes:

PB000001 esta placa tem uma única inserção de cada tipo de componente,

recaindo num caso de QAP trivial conforme apresentado na tabela 1.

PB000003 é uma variante do primeiro, onde sobre um percurso ótimo atribui-se componentes de tal sorte que nele nunca ocorra mais de uma troca de escaninho por inserção, fazendo que o conjunto de escaninhos faça um "vai e vem" nesse caminho ótimo.

Problema	Componentes	Escaninhos	Classe
PB000001	120	120	QAP trivial
PB000003	120	30	QAP trivial

Tabela 6: Casos de Teste Sintéticos

A configuração destes problemas está descrita na tabela 6. Na tabela 7, são exibidos os resultados para os casos sintéticos. Nesta tabela, indicam-se o tempo inicial obtido pelo embaralhamento aleatório de uma solução pré-definida, o tempo de inserção obtido com o OPTIMA, o tempo mínimo teórico, que neste caso é a configuração mínima definida pelo traçado sintético, e o percentual de melhoria obtido, neste caso já considerando o uso do algoritmo CRAFT modificado.

Placa	Inicial [s]	OPTIMA [s]	Lim. Inferior [s]	% Total
PB000001	80.15	47.4	42.7	40.8
PB000003	79.86	47.8	42.5	40.1

Tabela 7: Resultados Finais para os Casos de Teste Sintéticos

Observa-se também que o OPTIMA conseguiu baixar bem os tempos em relação à média das instâncias aleatórias, ficando porém mais de 10% acima do limite inferior ou da possível solução ótima. Entretanto, como o objetivo

principal do OPTIMA é o de obter boas soluções num tempo de processamento razoável, podemos considerar o resultado acima como sendo de boa qualidade.

7 CONCLUSÕES e PERSPECTIVAS

Como resultado deste trabalho, em particular das análises apresentadas no capítulo 6, pode-se afirmar que a técnica de A-Teams mostra-se apropriada para abordar problemas de otimização similares aos da otimização da máquina insersora AVK.

Os resultados mostram que a abordagem de combinar o QAP com o TSP consegue obter uma melhora adicional nos tempos de inserção de 4,7% em média em relação a uma abordagem que utiliza exclusivamente o TSP.

Como uma vantagem no uso desta técnica, podemos ressaltar sua inerente modularidade. O fato dos agentes nos A-Teams serem autônomos facilita a modularização desde a concepção até a implementação, sendo adaptada ao desenvolvimento incremental[33].

Uma vez determinado o problema e escolhidos os algoritmos, já se podem iniciar os experimentos, através da construção de um sistema mínimo, e ir incrementalmente ampliando-o conforme as necessidades.

Em particular, no caso do OPTIMA, pode-se experimentar a modificação de um algoritmo para resolução do QAP, para especializá-lo mais e melhorar o desempenho do sistema.

O uso de A-Teams para solução de QAPs, mesmo puros, é uma segunda contribuição deste trabalho, já que não há na literatura consultada e na página WWW da QAPBIB [BUR96] nenhuma menção ao uso dessa técnica.

[33] Também denominado desenvolvimento em espiral [BOE88].

Há, portanto, uma nova e interessante linha de pesquisas nessa área, mormente por ser um problema que tem se mostrado bem difícil de resolver. Uma possibilidade viável é a construção de A-Teams específicos para a resolução de instâncias do QAP, visando realizar uma avaliação da técnica. Poderia-se reduzir os limites inferiores conhecidos ao tratar instâncias dos problemas para os quais não se conheçam ainda os valores ótimos, de forma análoga ao trabalho realizado por [SOU93] para o TSP.

Outra frente de possibilidades, não explorada neste trabalho, consiste na criação de sistemas baseados na técnica de A-Teams que tratem dos problemas de otimização do nível 2 (seção 2.4), e dos problemas de recuperação discutidos na seção 3.1.

Numa abordagem ainda mais abrangente, poder-se-ia imaginar um sistema que englobasse outras fases do ciclo do projeto das placas de circuito impresso, recebendo os dados de entrada diretamente das ferramentas CAD de desenvolvimento das placas de circuito impresso. Assim, permitir-se-ia então a manipulação de mais de uma dimensão para a otimização do processo de manufatura, através da modificação do próprio layout das placas de circuito impresso. A adaptação dos A-Teams para múltiplas representações de um problema [KAN95] e a sua característica de poder tratar de problemas não necessáriamente disjuntos (seção 4.3) indicam a adequação desta técnica para se abordar essa classe de problemas.

Considerando possíveis melhorias na implementação do sistema OPTIMA, prevê-se o desenvolvimento do painel "Estatísticas", onde poder-se-á examinar o desempenho dos agentes vendo a progressão da melhora de solução para cada um deles, seu tempo de processamento, o número de soluções

contribuida por cada um deles, etc.

Bibliografia

[AHO83] Aho, A. V., Hopcroft, J. E. and Ullman, J. O. "Data Structures and Algorithms," Reading, Addison Wesley, 1983.

[ANS98a] Anstreicher, K. and Wolkowicz, H. "On Langrangian Relaxation of Quadratic Matrix Constraints," Research Report CORR 98-24, University of Waterloo, Dept. of Combinatorics and Optimization, Canada, 1998.

[ANS98b] Anstreicher, K.; Chen, X.; Wolkowicz, H.; Yuan, Y. "Strong Duality for Trust-Region Type Relaxation of the Quadratic Assignment Problem," Research Report CORR 98-31, University of Waterloo, Dept. of Combinatorics and Optimization, Canada, 1998.

[ARI90] Ben-Arieh, D. and Dror, M. "Part Assignment to Electronic Insertion Machines: two machine case," *International Journal of Production Research*, Vol. 28, Nº 7, 1317-1327, 1990.

[ARM63] Armour, G. C. and Buffa, E. S. "A Heuristic Algorithm and Simulation Approach to the Relative Location of Facilities," *Managemente Science*, 9, 294-309, 1963.

[BAL88] Ball, M. O. and Magazine M. J. "Sequencing of Insertions in Printed Circuit Board Assemblies," *Operations Research*, Vol. 36, 192-201 (1988).

[BAT94] Battiti, R.; Tecchiolli, G. "The Reactive Tabu Search," *ORSA Journal on Computing*, Vol. 6, 126-140, 1994.

[BLO92] Bloomer, J. "Power Programming with RPC," Sebastopol, O´Reilly & Associates, Inc., 1992.

[BOE88] Boehm, B. W. "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, 61-72, maio de 1988.

[BRO96] Broad, K.; Mason, A.; Rönnqvist, M. and Frater, M. "Optimal Robotic Component Placement," *Journal of the Operational Research Society*, Vol. 47, 1343-1354, 1996.

[BUR96] Burkard, R. E., Karish, S. R. e Rendl, F. "QAPLIB-A Quadratic Assigment Problem Library". *European Journal of Operational Research*, 55:115-119, 1991.

[CAR80] Carpaneto, G. and Toth, P. "Algorithm 548 Solution of the Assigment Problem [H]," *ACM Transactions on Mathematical Software*, Vol. 21, Nº 4, 410-415, 1980.

[CAR95] Carpaneto, G.; Dell'Amico, M.; Toth, P. "Algorithm 750: CDT: A Subroutine for the Exact Solution of Large-Scale, Asymmetric Traveling Salesman Problems," *ACM Transactions on Mathematical Software*, Vol. 21, Nº 4, 410-415, 1995.

[CHA87] Chang, T., Terwillinger Jr., J. "A rule based system for printed wiring assembly process planning" *International Journal of Production Research*, Vol. 25, Nº 10, 1465-1482, 1987.

[CHA89] Chan, D. And Mercier, D. "IC Insertion: An Application of the Traveling Salesman Problem" *International Journal of Production Research*, Vol. 27, Nº 10, 1837-1841, 1989.

[CLA64] Clarke, G. and Wright, J. W. "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Operations Reseach*, Vol. 12, 568-581, 1964.

[CON70] Converse, A. O., "Otimização", Trad. de Altair Rios Neto, São Paulo, EDART - São Paulo Livraria Editora Ltda., 1977.

[COO67] Coombs, C. F., Coombs Jr., C. F., "Printed Circuits Handbook," New York, McGraw Hill, 1967.

[CRA90] Crama, Y. e Spieksma, F. C. R. "Throughput Rate Optimization in the Automated Assembly of Printed Circuit Boards", *Annals of Operational Research*, 26, 455-480, 1990.

[CRA96a] Crama, Y. et al., "The Assembly of Printed Circuit Boards: a case with multiple machines and multiple board types", *G.E.M.M.E.*, Nº 9615, Juillet 1996.

[CRA96b] Crama, Y. et al., "The Component Retrieval Problem in Printed Board Assembly", *The International Journal of Flexible Manufacturing Systems*, 8, 287-312, 1996.

[CHR89] Christofides, N.; Benavent, E. " An Exact Algorithm for the Quadratic Assignment Problem," *Operations Research*, Vol. 37, 760-768, 1989.

[CRO58] Croes, G. A., "A Method for Solving Traveling Salesman Problems," *Operations Reseach*, Vol. 6, 791-812, 1958.

[CUN95] Cun, B. le; Roucariol, C. and the PNN Team "BoB: A Unified Platform for Implementing Branch and Bound like Algorithms" PRiSM Rapports de Recherche no 95/16, 1995.

[DAN54] Dantzig, G. B., Fulkerson, D. R. and Johnson, S. M., "Solution of a Large-Scale Traveling-Salesman Problem," *Operations Research*, 2, 393-410, 1954.

[DAV91] "Handbook of Genetic Algorithms," Davis, L. editor, Van Nostrand Reinhold, 1991.

[EVA87] Evans, J. R., "Structural Analysis of Local Search Heuristics in Combinatorial Optimization," *Computer Operations Research*, Vol. 14, No 6, 465-477, 1987.

[FIN75] Fink, D. G., Editor-in-Chief, "Electronics Engineers Handbook," New York, McGraw Hill, 1975.

[FRI79] Frieze, A. M., "Worst-case Analysis of Algorithms for Traveling Salesman Problems," *OR Verfahen*, 32, 93-112, 1979.

[GAN94] Ganski, C. Y., "Analysis of Palubetskis's Algorithm," `http://xfactor.wpi.edu/Works/DC/qap/qapfin94`, 1994.

[GAR79] Garey, M. R. and Johnson, D. S., "Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completness," San Francisco, Freeman, 1979.

[GLO89] Glover, F. "Tabu Search - Part I," *ORSA Journal Of Computing*, Vol. 1 No 3, 190-206, 1989.

[GLO90] Glover, F. "Tabu Search - Part II," *ORSA Journal Of Computing*, Vol. 2, No 1, 4-32, 1990.

[GOL89] Golany, B and Rosenblatt, M. J. "A Heuristic Algorithm for the Quadratic Assignment Formulation to the Plant Layout Problem,", *International Journal of Production Research*, Vol. 27, no 2, 293-308, 1989.

[GOU99] Gouveia, L. and Pires, J. M. "The Asymmetric Traveling Salesman Problem and a Reformulation of the Miller-Tucker-Zemlim Constrains," *European Journal of Operational Research*, 112:134-146, 1999.

[HAD98a] Hadley, S. W.; Rendl, F.; Wolkowicz, H. "A New Lower Bound via Projections for the Quadratic Assignment Problem," Artigo não publicado. Disponível na página WWW da QAPLIB.

[HAD98b] Hadley, S. W.; Rendl, F.; Wolkowicz, H. "Bounds for the Quadratic Assignment Problem Using Continuous Optimization Techniques," Artigo não publicado. Disponível na página WWW da QAPLIB.

[HAH98] Hahn, P.; Grant, T.; Hall, N. "A Branch-and-Bound Algorith for the Quadratic Assignment Problem Based on the Hungarian Method," *European Journal of Operational Research*, 108:629-640, 1998.

[HAN72] Hanan, M. and Kurtzberg, J. M. "A Review of the Placement and Quadratic Assignment Problems," *SIAM Review*, Vol. 14, 324-342, 1972.

[HEL70] Held, M. and Karp, R. M. "The Traveling-Salesman Problem and Minimum Spanning Trees," *Operations Reseach*, Vol. 18, 1138-1162, 1970.

[HOU90] Houshyar, A.; McGinnis, L. F. "A Heuristic for Assigning Facilities to Locations to Minimize WIP Travel Distance in a Linear Facility," *International Journal of Production Engineering*, Vol. 28, Nº 8, 1485-1498, 1990.

[IBM96a] Rachlin, J.; Wu, F.; Murthy, S.; Talukdar, S.; Sturnzenbecker, M.; Akkiraju, R.; Fuhrer, R.; Aggarwal, A.; Yeh, J. Henry, R.; Jaramaraman, R. "Forest View: A System for Integrated Scheduling in Complex Manufacturing Domains," IBM Report, 1996.

[IBM96b] Lee, H.; Murthy, S.; Haider W.; Morse, D., "Primary Production Scheduling At Steel Making Industries," *IBM Journal Of Research And Development*, Vol. 40, Nº 2, 231-252, 1996.

[KAN95] Kang, T. C. "Solving Train Scheduling Problems Using A-Teams," Ph. D. Dissertation, Department Of Electrical And Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, 1994.

[KAN97] Kang, T. C. And Talukdar, S. "Overview on Designing Autonomous Agents Teams (A-Teams)". /Apresentado na Conference on Management and Control of Production and Logistics MCPL'97, Campinas, 1997/

[KAP94] Skorin-Kapov, J. "Extensions of a Tabu Search Adaptation to the Quadratic Assignment Problem," *Computer Operations Research*, Vol. 21, Nº 8, 855-865, 1994.

[KAR93] Karisch, S. E.; Rendl, F.; Wolkowicz, H. "Trust Regions and Relaxations for the Quadratic Assignment Problem," Simultaneamente emitido como Research Report CORR 93-15, University of Waterloo, Dept. of Combinatorics and Optimization, Canada, 1998; Report SOR 93-10, Princenton University, Dept. of Civil Engineering and Operations Research, e Report 93-34, DIMACS Center, Rutgers University.

[KIR83] Kirkpatrick, S.; Gellat, C. D.; Cecchi, M. P., "Optimization by Simulating Annealing," *Science*, Vol. 220, Nº 4598, 1983.

[KOO57] Koopmans, T. C. and Beckman, M. "Assignment Problems and the Location of Economic Activities," *Econometrica*, Vol. 25, 53-76, 1957.

[LAC93] Lacksonen, T. A.; "Quadratic Assignment Algorithms for the Dynamic Layout Problem," *Intl. Journal Production Research*, Vol. 31, Nº 3, 503-517, 1993.

[LAW63] Lawler, E. L., "The Quadratic Assigment Problem," *Management Science*, Vol. 9, 586-599, 1963.

[LAW85] Lawler, E. L. et al. (ed.) "The Traveling Salesman Problem" John Wiley & Sons, Chichester, 1985.

[LEI89] Leipälä, T.; Nevalainen, O. "Optimization of the Movements of a Component Placement Machine," *European Journal of Operational Research*, 38, 167-177, 1989.

[LEW81] Lewis, H. R.; Papadimitriou C. H. "Elements of the Theory of Computation," New Jersey, Prentice Hall, Inc., 1981.

[LIN65] Lin, S. "Computer Solutions of the Traveling Salesman Problem," *Bell Systems Technical Journal*, Vol. 44, 2245-2269, 1965.

[LIN73] Lin, S.; Kernighan, B. W. "An Effecttive Heuristic Algorithm for the Traveling Salesman Problem," *Operations Reseach*, Vol. 21, 498-516, 1973.

[MAI91] Maimon, O. e Shtub, A. "Grouping Methods for Printed Circuit Board Assembly," *International Journal of Production Research*, Vol. 29, Nº 7, 1379-1390, 1991.

[MAN95] Mans, B.; Mautor, T.; Roucariol, C. "A Parallel Depth First Search Branch and Bound Algorithm for the Quadratic Assigment Problem," *European Journal of Operational Research*, 81:617-628, 1995.

[MCG92] McGinnis, L. F.; Ammons, J. C.; Carlyle, M.; Crammer, L.; Depuy, G. W.; Ellis, K. P.; Tovey, C. A.; Xu, H., "Automated Process Planning for Printed Circuit Card Assembly," *IIE Transactions*, Vol. 24, Nº 4, 18-29, 1992.

[MIC93] "Microsoft Win32 API Programmer´s Reference," s.l., Microsoft Corp., 1993.

[MIL60] Miller, C. E., Tucker, A. W. and Zemlin, R. A., "Integer Programming Formulations and Traveling Salesman Problems," *Journal ACM*, 7, 326-329, 1960.

[MIY77] Miyake, M. Y. "The Quadratic Assignment Model Applied to Solve Facility Location Problems," Ph. D. Thesis, University Of London, London School Of Economics, 1977.

[NAS96] Nascimento, H. A. D, Mendonça Neto, C. F. X. e Souza, P. S. "Sinergia em Desenho de Grafos Usando Springs e Pequenas Heurísticas", Technical Report IC-96-15, Unicamp, Campinas, 1994.

[NII96] Nii, H. P. "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures, Parts I And II," *AI Magazine*, 7:2 E 7:3, 1986.

[NUG68] Nugent, C. E., Vollmann, T. E. and Ruml, J. "An Experimental Comparision of Techniques for the Assignment of Facilities to Locations," *Operations Reseach*, Vol. 16, 150-173, 1968.

[PAN96] Panasonic. "Catálogo Panasert Electronic Component Placement/Insertion Machines," Matsushita Electric Industrial Co., Ltd., 27, 1996.

[PAR94] Pardalos, P.; Rendl, F.; Wolkowicz, H. "The Quadratic Assignment Problem: A Survey and Recent Developments." In: Pardalos, P.; Rendl, F.; Wolkowicz (Eds.), *Proceedings of the DIMACS Workshop on Quadratic Assignment Problems*, DIMACS Series in Discrete Mathematics vol 16, 1-42, 1994.

[PAT90] Patterson, D. A.; Hennessy, J. L. II "Computer Architecture a Quantitative Approach," Morgan Kauffmann Publishers, Inc., San Mateo, 1990.

[PEI94a] Peixoto, H. e Souza, P. "Times Assíncronos: Uma Nova Técnica para o *Flow Shop Problem*", Technical Report DCC-94-06, Unicamp, Campinas, 1994.

[PEI94b] Peixoto, H. e Souza, P. "Uma Metodologia de Especificação de Times Assíncronos", Technical Report DCC-94-12, Unicamp, Campinas, 1994.

[POT93] Potvin, J. Y. "The Traveling Salesman Problem: A Neural Network Perspective," *ORSA Journal On Computing*, 5, 328-348, 1993.

[PRE92] Press, W. H.; Vetterling, W. T.; Teukolsky, S. A.; Flannery, B. P. "Numerical Recipes in C, the Art of the Scientific Computing," Cambridge, Cambrige Univ Press, 1992.

[QUA91] Quadrel, R. W. "Asynchronous Design Environments: Architecture and Behaviour". Ph. D. Dissertation, Department of Architecture, College Of Fine Arts, Carnegie Mellon University, Pittsburgh, PA, 1991.

[RAM98] Ramachandran, B. and Pekny, J. F. "Lower Bounds for Nonlinear Assignment Problems Using Many Body Interactions," *European Journal of Operational Research*, 105:202-215, 1998.

[RAR98] Rardin, R. L. "Optimization In Operations Research" Prentice Hall, Inc., Upper Saddle River, NJ, 1998.

[REI94] Reinelt, G. "The Traveling Salesman Computational Solutions for TSP Applications" Springer-Verlag, Berlin, 1994.

[RES95] Resende, M. G. C.; Ramakrishnan, K. G.; Drezner, Z. "Computing Lower Bounds for the Quadratic Assignment Problems with an Interior Point Algorithm for Linear Programing," *Operations Research*, Vol. 43, 781-791, 1995.

[RES96] Resende, M. G. C.; Pardalos, P. M.; Li, Y. "Algorithm 754: Fortran Subroutines for Appoximate Solution of Dense Quadratic Assignment Problems Using GRASP," *ACM Transactions on Mathematical Software*, Vol. 22, Nº 1, 104-118, 1996.

[ROD95] Rodrigues, R. e Souza, P. "Asynchronous Teams: A Multi-Algorithm Approach for Solving Combinatorial Multiobjective Optimization Problems", , Technical Report DCC-95-18, Unicamp, Campinas, 1995.

[SAH76] Sahni, S.; Gonzalez, T. "P-Complete Approximation Problems," *J. Assoc. Comput. Mach.*, Vol. 23, 555-565, 1976.

[SAV80] Savulescu, S. C. "Grafos Dígrafos e Redes Elétricas Aplicações na Pesquisa Operacional" IBEC, São Paulo, 1980.

[SCA70] Scarlett, J. A., "Printed Circuit Boards for Microelectronics," New York, Van Nostrand Reinhold Co., 1970.

[SHE86] Shevell, S. F.; Buzacott, J. A.; Magazine, M. J. "Simulation and Analysis of a Circuit Board Manufacturing Facility," *Proceedings Of The 1986 Winter Simulation Conference* (December), Washignton, DC, 1986.

[SPI71] Spivey, W. A., "Introdução à Programação Linear", Trad. de Amâncio F. Pulcherio, São Paulo, Cia. Editora Nacional, 1975.

[SOU93] Souza, P. "Asynchronous Organizations For Multi-Algorithm Problems," Ph. D. Dissertation, Dept. of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, 1993.

[SOU93a] Souza, P.; Talukdar, S. "Asynchronous Organizations For Multi-Algorithm Problems," *ACM Symposium on Applied Computing*, Indianopolis, 1993.

[STE61] Steinberg, L., "The Backboard Wiring Problem: a placement algorithm," *SIAM Review*, Vol. 3, 37-50, 1961.

[STE70] Steckhan, H. "A Theorem on Symmetric Traveling Salesman Problems," *Operations Reseach*, Vol. 18, 1163-1167, 1970.

[TAH97] Taha, H. A. "Operations Research : an introduction" Prentice Hall, Inc., Upper Saddle River, NJ, 1997.

[TAL96] Talukdar, S.; Baerentzen, L.; Gove, A.. Souza, P. "Asynchronous Teams: Cooperation Schemes For Autonomous Agents," Engineering Design Center, Carnegie Mellon University, 1996.

[TAL98] Talukdar, S. "Rules for Collaboration among Cyber-Agents" Invited Talk, Informs National Meeting, Montreal, Quebec, April 26-28, 1998.

[TAM95] Tamura, R. A., ed. "Programming Windows 95 Unleashed," Sams Publishing, Indianapolis, 1995.

[VAL97] Valenzuela, C. L and Jones, A. J. "Estimating the Held-Karp Lower Bound for the Geometric TSP," *European Journal of Operational Research*, 102:157-175, 1997.

[VIA98] Viana, G. V. R., "Meta-Heurísticas e Programação Paralela em Otimização Combinatória", Fortaleza, Edições UFC, 1998.

[WAL84] Walas, R. A. and Askin, R. G. "An algorithm for NC Turret Punch Press Tool Location and Hit Sequencing," *IEE Transactions*, Vol. 16, Nº 3, 280-287, 1984.

[WAM96] Wampler, B. E., "V: A Portable GUI Framework," *C/C++ Users Journal*, Vol. 14, Nº 6, 1996.

[WES83] West, D. H. "Algorithm 608 Approximate Solution of the Quadratic Assignment Problem," *ACM Transactions on Mathematical Software*, Vol. 9, Nº 4, 461-466, 1983.

[ZHA98] Zhao, T. C. and Overmars, M. "Forms Library A Graphical User Interface Toolkit for X V0.88.1," User's Manual, 1998.

[ZHA98b] Zhao, Q.; Karisch, S. E.; Rendl, F.; Wolkowicz, H. "Semidefinite Programmin Relaxations for the Quadratic Assignment Problem," Versão revisada emitida em 1998 do relatório técnico Report CORR 95-27, University of Waterloo, e DIKU TR-96/32, University of Copenhagen. Disponível em `ftp://orion.uwaterloo.ca/pub/henry/reports/ABSTRACTS.html`.

A Formato do arquivo de programa de inserção

Cada linha do arquivo de programa de inserção da máquina AVK contém as seguintes informações:

- posição do componente na placa através de suas coordenadas X e Y;
- tipo de componente especificado de forma indireta através do escaninho de onde deve ser retirado o componente a inserir;
- informações adicionais para processamento, tais como ângulo para posicionar corretamente o componente, distância da dobra dos fios, etc.

A primeira e última linha do arquivo servem de cabeçalho e marcador de fim de programa de inserção não portando informação de inserção de componentes. As linhas intermediárias (os "blocos") têm a seguinte estrutura:

```
N0001/0G1M000T004X+000000Y+000000Z+00000V+00000W+00000       +00000D
^    ^ ^ ^   ^   ^       ^       ^   ^      ^      ^
|    | | |   |   |       |       |   |      |      |
|    | | |   |   |       |       |   |      |      |
|    | | |   |   |       |       |   |      |      |
1    2 3 4   5   6       7       8   9     10     11
```

1 nº do bloco (linha)

2 salto de bloco

3 não documentado

4 não documentado

5 Ângulo de inserção

6 Coordenada X

7 Coordenada Y

8 Escaninho

9 Altura da inserção

10 Largura da dobra (pitch)

11 não documentado

O formato típico do arquivo é o seguinte:

```
N0001/0G1M000T004X+000000Y+000000Z+00000V+00000W+00000      +00000D
N0002/0G0M001T002X+006465Y+007088Z+00001V+00000W+00732      +00000D
N0003/0G0M001T002X+007746Y+003362Z+00002V+00000W+00738      +00000D
N0004/0G0M001T002X+008405Y+003378Z+00002V+00000W+00736      +00000D
.
.
.
N0247/0G0M001T002X+020209Y+003379Z+00002V+00000W+00734      +00000D
N0248/0G0M001T002X+019551Y+003363Z+00002V+00000W+00732      +00000D
N0249/0G0M001T002X+018271Y+007088Z+00001V+00000W+00743      +00000D
N0250/0G0M000T000X+000000Y+000000Z+00000V+00000W+00000      +00000D
*
```

B Script para gerar matrizes do QAP

O script AWK maq2qap.awk analisa um arquivo de programa de inserção da AVK e gera as duas matrizes, de fluxo e de distância, associadas ao QAP correspondente.

```
#!/usr/bin/awk -f

# Este script gera um problema QAP na forma can{\^o}nica:
#  tamanho do problema (ordem das matrizes);
#  uma "matriz de dist{\^a}ncias";
#  e uma "matriz de fluxo";
# a partir dum arquivo "m{\'a}quina" de insersora de componentes
# AVK do Exclui os blocos contendo o c{\'o}digo de salto "/7"


# Para usar este script use a seguinte linha de comando:
#
# gawk -f fluxo.awk ARQUIVO_MAQUINA.NCD
#
# onde:
#
# ARQUIVO_MAQUINA.NCD :: arquivo de entrada no formato "m{\'a}quina" Panasert
#

# Linha t{\'\i}pica do arquivo Panasert:
#
# N0001/0G1M000T004X+000000Y+000000Z000V+00000W+00000                    +00000D
#

BEGIN {                          # Iniciar vari{\'a}veis
    EscAnt = 0;
    EscMax = 0;
    EscTempo[0] = 0;             # Usamos inteiros para facilitar os testes
    EscTempo[1] = 33;
    EscTempo[2] = 37;
    EscTempo[3] = 41;
    EscTempo[4] = 43;
    EscTempo[5] = 44;
```

```
    EscTempo[6] = 9999;          # Este tempo p/usar p/qqr troca acima de
                                 # seis escaninhos (oper. AVK)
}

{ FIELDWIDTHS = "1 4 1 1 1 1 1 3 1 3 1 7 1 7 1 6 1 6 1 6 15 6 1"

# Com as defini{\c c}{\~o}es acima, ficamos com os seguintes campos:
#
#  $2 :: n{\'u}mero de bloco
# $10 :: {\^a}ngulo de inser{\c c}{\~a}o
# $12 :: coordenada X x 100
# $14 :: coordenada Y x 100
# $16 :: escaninho
# $20 :: largura  dobra (pitch)
#

  if ($2 > 1 && $0 !~ /\*/ && $4 !~ 7) { # linha for bloco v{\'a}lido
      if (EscAnt) {
  Flow[EscAnt + 0,$16 + 0]++;
      }
      EscAnt = $16 + 0;
      if ($16 > EscMax){
  EscMax = $16 + 0;
      }
  }
}

# Gera{\c c}{\~a}o QAP
END {
  printf("%s\n\n", EscMax);              # Tamanho do QAP
    for (i = 1; i <= EscMax; i++) {      # Imprime a matriz fluxo
for (j = 1; j <= EscMax; j++)
    printf("%02d ", i == j? 0:Flow[i,j]);
printf("\n");
    }
    printf("\n");
    for (i = 1; i <= EscMax; i++) {      # Imprime a matriz dist{\^a}ncias
      for (j = 1; j <= EscMax; j++) {
    troca = i > j?i - j:j - i;
    printf("%04d ", EscTempo[troca > 5? 6:troca]);
      }
```

```
printf("\n");
    }
    printf("\n");
}
```