OTIMIZAÇÃO DO POSICIONAMENTO E DA MONTAGEM AUTOMÁTICA DE COMPONENTES ELETRÔNICOS ATRAVÉS DE UM ALGORITMO GENÉTICO

Carlos Henrique Craveiro da Fonseca

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

_____

Prof. Amit Bhaya, Ph.D.

_____

Prof. Eugenius Kaszkurewicz, D.Sc.

_____

Prof. Valmir Carneiro Barbosa, Ph.D.

_____

Prof. Wilhelmus Adrianus Maria Van Noije, Ph.D.

RIO DE JANEIRO, RJ – BRASIL

ABRIL DE 2005

FONSECA, CARLOS HENRIQUE CRAVEIRO DA

Otimização do Posicionamento e da Montagem Automática de Componentes Eletrônicos através de um Algoritmo Genético [Rio de Janeiro] 2005

XII, 80p, 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2005)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Otimização
2. Algoritmos Genéticos

I. COPPE/UFRJ II. Título (série)

À minha adorável Ana Paula, Henriquinho, pais e irmã pelo amor mais precioso.

Agradecimentos

Este trabalho não teria sido concluído não fosse o imenso apoio de minha família nas horas em que precisei abrir mão do convívio para dedicar-me à ciência e à pesquisa.

Agradeço aos meus orientadores Dr. Amit Bhaya e Dr. Eugenius Kaszkurewicz e aos coordenadores do curso de Mestrado Dra. Marli Costa (NUTELI/UFAM) e Dr. Ramon Romankevicius (COPPE/UFRJ) por acreditarem, primeiramente, na proposta pioneira e audaciosa de realizar um curso de mestrado em Engenharia Elétrica em Manaus e pela inestimável colaboração no desenvolvimento desta tese. O apoio, incentivo e amizade foram fundamentais nos momentos mais difíceis.

A realização dos testes práticos em máquina só foi possível graças ao apoio da empresa SONY Brasil ltda. e empenho dos seus técnicos programadores de máquina, Juarez Silva e Virgínia Miranda, que não mediram esforços em contornar todos os obstáculos.

Não poderia deixar de agradecer ao Sr. Yasuhiro Tsujimura do Nippon Institute of Technology por ter-me, gentilmente, cedido cópia impressa de toda sua obra, complementando a bibliografia desse trabalho.

Gostaria, ainda, de manifestar meus sinceros agradecimentos à SUFRAMA pelo suporte financeiro para a realização deste curso de mestrado.

Para mim, implementar esse trabalho de pesquisa e escrever a dissertação foi não só a concretização de um antigo sonho, como também uma prova de que não existem obstáculos impostos pelo tempo que a dedicação, empenho e entusiasmo não possam vencer.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc)

OTIMIZAÇÃO DO POSICIONAMENTO E DA MONTAGEM AUTOMÁTICA DE COMPONENTES ELETRÔNICOS ATRAVÉS DE UM ALGORITMO GENÉTICO

Carlos Henrique Craveiro da Fonseca

Abril/2005

Orientadores: Amit Bhaya
Eugenius Kaszkurewicz

Programa: Engenharia Elétrica

Este trabalho propõe uma solução original para o problema de otimização da montagem de componentes eletrônicos em máquinas SMT Multi-Head utilizando a técnica de algoritmos genéticos. Tomou-se o caso prático de uma máquina JUKI modelo KE-2030 para efeito de modelagem, implementação, testes e validação. Duas importantes vantagens do algoritmo proposto sobre os existentes na literatura são a otimização simultânea do roteamento e alocação de alimentadores, bem como a possibilidade de utilização de nozzles repetidos nos cabeçotes múltiplos. Os resultados obtidos revelam que o algoritmo desenvolvido apresenta eficiência comparável a do software comercial HLC utilizado na indústria.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

OPTIMIZATION OF AUTOMATIC ROUTING AND PLACEMENT OF ELECTRONIC COMPONENTS USING A GENETIC ALGORITHM.

Carlos Henrique Craveiro da Fonseca

April/2005

Advisors: Amit Bhaya
Eugenius Kaszkurewicz

Department: Electrical Engineering

This work proposes an original solution for the optimization problem of electronics components assembly in a Multi-Head SMT placement machine using genetic algorithms. For modeling, implementation, tests and validation purposes, the practical case of a JUKI KE-2030 machine was chosen. Two important advantages of the algorithm developed in this thesis, with respect to existing algorithms, are that it permits the usage of repeated nozzles in the multiple machine heads and, furthermore, that it simultaneously optimizes both the routing and feeder assignment problems. The results show that the efficiency of the developed algorithm is comparable to that of the HLC commercial software used in industry.

Índice

	CAPÍTULO	Página
1.	**Introdução**	1
	1.1 Caracterização do Problema	5
	1.2 Objetivos	8
	1.3 Organização do texto	8
	1.4 Recursos computacionais	9
2.	**Formulação Matemática e Revisão Bibliográfica**	10
	2.1 Caso *Single-Head* (SH)	13
	2.2 Caso *Multi-Head* (MH): modelo utilizando o conceito de Hiperaresta e Hipergrafo	15
	2.3 Caso *Multi-Head* (MH): formulação como problema de Programação Matemática	17
	2.3 Soluções para o problema utilizando Métodos Aproximados	21
3.	**Algoritmo Genético Proposto**	25
	3.1 Caso *Single-Head* (SH)	26
	3.1.1 Formação dos Cromossomos	26
	3.1.2 Função Objetivo	28
	3.1.3 Operadores do AG	30
	3.2 Caso *Multi-Head* (MH)	33
	3.2.1 Formação dos Cromossomos - MH	33
	3.2.2 Utilização de Nozzles Repetidos - Caso MH-NR	36
	3.2.3 Função Objetivo para o caso MH	38
	3.2.4 Problemas com a abordagem simultânea – TAP, RAP e PAPSP	41
	3.2.5 Inclusão de componentes vazios (‘*blank*’ *components*)	42

3.2.6 Desacoplamento do problema de *nozzle assignment* - TAP 44

4. Resultados 49

4.1 Avaliação do tempo de execução do AG 49

4.2 Dados do problema exemplo 51

4.3 Resultados do HLC 52

4.3.1 HLC: Placement & Nozzle Assignment 53

4.3.2 HLC: Feeder Assignment 56

4.4 Resultados do AG 57

4.5 Análise Comparativa das Soluções 60

5. Conclusões e Pesquisa Futura 62

Referências Bibliográficas 66

Apêndice

O que é um AG ? 70

Soluções do TSP utilizando AGs (Metaheurística) 71

Operadores Genéticos 71

Cruzamento 72

Mutação 73

Função Objetivo e Função de Aptidão 75

Seleção 77

Método da Roleta 78

Método SUS (Stochastic Universal Sampling) 79

Índice de figuras:

Figura 1.1 –	Exemplo de Configuração de Máquinas em uma Linha de Montagem de SMD	1
Figura 1.2 –	Comparação dos processos de montagem com pasta e adesivo	2
Figura 1.3 –	Evolução da Tecnologia de Montagem de Comp. Eletrônicos	3
Figura 1.4 –	Tendência tecnológica dos CIs (*Downsizing x Increase in Pins*)	4
Figura 1.5 –	Tipos de Máquina SMT	5
Figura 1.6–	Máquina JUKI modelo KE-2030	6
Figura 1.7–	Vista superior (plano) de uma Máquina *Multi-Head* [5]	6
Figura 2.1 –	Métodos de Otimização Combinatória	23
Figura 2.2 –	Caract. de Máquinas e trabalhos encontrados na literatura	24
Figura 3.1 –	Estrutura do LinkA	27
Figura 3.2 –	Estrutura do LinkB	28
Figura 3.3 –	Diagrama da Função Objetivo	28
Figura 3.4 –	Percurso para componentes que utilizam o mesmo *nozzle*	29
Figura 3.5 –	Percurso para componetes que utilizam *nozzles* diferentes	30
Figura 3.6 –	Operador de Mutação (Inversão)	31
Figura 3.7 –	Operador de Mutação (Troca Recíproca)	31
Figura 3.8 –	Operador de Mutação (Heurístico)	31
Figura 3.9 –	Cruzamento Ordenado	32
Figura 3.10 –	Estrutura do LinkA para o caso *Multi-Head*	34
Figura 3.11 –	Estrutura dos Links B e B1 para o caso *Multi-Head*	34
Figura 3.12 –	Exemplo do que representam os Links B e B1	35
Figura 3.13 –	Associação de *nozzles* aos cabeçotes	35

Figura 3.14 – *Pickup* simultâneo entre cabeçotes h1 e h2 39

Figura 3.15 – Exemplo de cromossomo solução (Link B) 42

Figura 3.16 – Arranjo (I) de *nozzles* por cabeçote do exemplo da fig. 4.15 42

Figura 3.17 – Arranjo (II) de *nozzles* por cabeçote do exemplo da fig. 4.15 43

Figura 3.18 – Exemplo de cromossomo acrescido de componentes vazios 43

Figura 3.19 – Transformaçào do *LinkN* em $\tilde{l}^{o}$ 46

Figura 3.20 – Estrutura do AG-TAP 46

Figura 4.1 – Resultado após a vetorização do AG 50

Figura 4.2 – Análise de tempos do AG 50

Figura 4.3 – Disposição física dos componentes na PCI 52

Figura 4.4 – Programa de montagem otimizado pelo HLC (parte I) 53

Figura 4.5 – Percursos com acionamento sequencial e otimizado 55

Figura 4.6 – Programa de montagem otimizado pelo HLC (parte II) 56

Figura 4.7 – Solução No. 1 do AG para o exemplo da tabela 5.1 58

Figura 4.8 – Solução No. 2 do AG para o exemplo da tabela 5.1 58

Figura 4.9 – Solução No.3 do AG para o exemplo da tabela 5.1 59

Figura 4.10 – Solução No.4 do AG para o exemplo da tabela 5.1 59

Figura 4.11 – Solução No.5 do AG para o exemplo da tabela 5.1 60

Figura A.1 – Estrutura Geral dos Algoritmos Genéticos 70

Figura A.2 – Exemplo de Roleta 78

Figura A.3 – Roleta utilizada no método de seleção SUS 79

Índice de Tabelas:

Tabela 3.1 –	Mapa de Alimentação	27
Tabela 3.2 –	Exemplo de um Programa de Montagem	28
Tabela 3.3 –	Exemplo de Lista de Componentes	37
Tabela 4.1 –	B.O.M e tipos de *nozzle* para o problema exemplo	52
Tabela 4.2 –	Representação dos dados do programa otimizado pelo HLC (parte I)	54
Tabela 4.3 –	Representação dos dados do programa otimizado pelo HLC (partes I e II)	56
Tabela 4.4 –	Análise comparativa das soluções propostas pelo AG e HC	61

Nomenclatura:

HLC - Algoritmo de otimização comercial desenvolvido para máquinas JUKI (*Host Line Computer*)

MH – Tipo de máquina com múltiplos cabeçotes (*Multi-Head*)

PAPSP - Problema de Roteamento de Componentes (*Pick and Place Problem*)

PCI - Placa de Circuito Impresso

RAP - Problema de configuração dos carretéis nos *slots* da máquina (*Reel Assignment Problem*)

SH – Tipo de máquina com único cabeçote (*Single-Head*)

SMD – Dispositivo de Montagem em Superfície (*Surface Mount Device*)

SMT – Tecnologia de Montagem em Superfície (*Surface Mount Technology*)

TAP - Problema de alocação de *nozzles* nos cabeçotes (*Tool Assignment Problem*)

TSP - Problema do Caixeiro Viajante (*Traveling Salesman Problem*)

Capítulo 1

Introdução

A tecnologia de montagem de componentes eletrônicos em superfície (SMT – *Surface Mount Technology*), utilizada na produção de placas de circuito impresso que compõem os mais diversos aparelhos eletrônicos, popularizou-se desde a última década, permitindo a miniaturização do produto final, o aumento da sua complexidade tecnológica e a melhoria de eficiência do processo produtivo.

A figura 1.1 ilustra uma típica configuração de máquinas utilizadas no processo industrial de montagem de componentes SMD (*Surface Mount Device*) sobre placas de circuito impresso. Este processo, com **pasta de solda**, compreende as etapas de: aplicação da pasta nas ilhas de cobre da placa de circuito impresso, **posicionamento e montagem automática dos componentes eletrônicos**, fusão da pasta de solda com os terminais dos componentes em um forno com perfil térmico controlado (*reflow oven*) e inspeção visual com câmera.

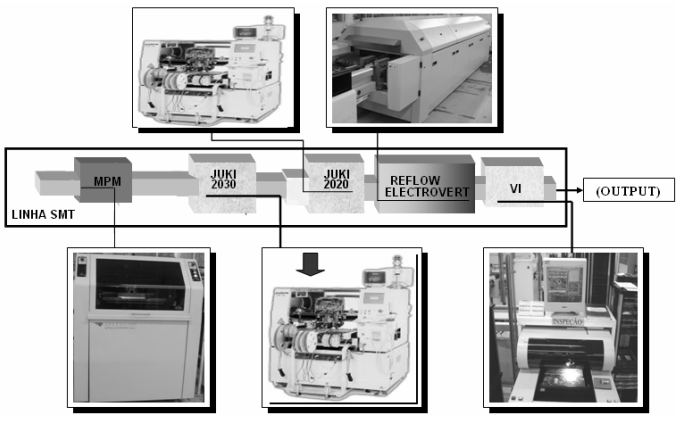


Figura 1.1 – Exemplo de Configuração de Máquinas em uma Linha de Montagem de SMD

Uma outra configuração de processo de montagem de componentes SMD utiliza, ao invés de pasta, **adesivo,** cujas etapas são: aplicação de adesivo nas posições de montagem dos componentes, **posicionamento e montagem automática dos componentes eletrônicos,** cura do adesivo (para fixar o componente na placa) em um forno com perfil térmico controlado (*reflow oven*) e soldagem dos componentes nas ilhas de cobre da placa em um forno de soldagem por onda (*wave soldering*). Esse processo normalmente é utilizado nas placas que utilizam componentes tipo '*through-hole*' (cujos terminais atravessam orifícios da placa) e discretos (radiais ou axias) além dos componentes SMD.

A figura 1.2 compara os dois processos de montagem de componentes SMD com pasta de solda e adesivo.

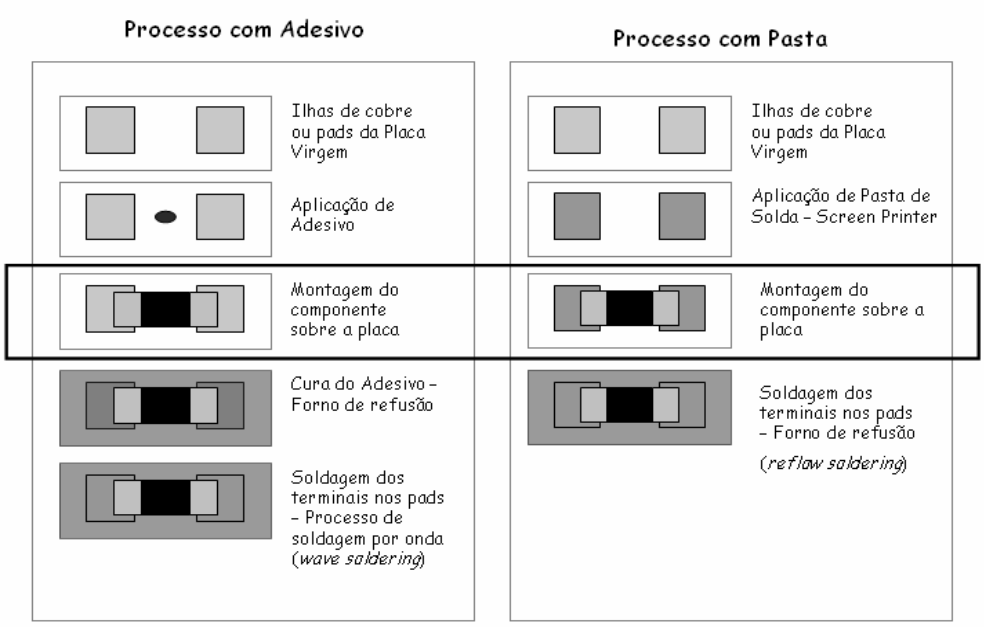


Figura 1.2 – Comparação dos processos de montagem com pasta e adesivo

O foco deste trabalho é a otimização da etapa de **posicionamento e montagem automática de componentes**, de forma a minimizar o ciclo de máquina considerado o gargalo da cadeia produtiva [1]. O impacto disso é significativo na melhora da produtividade e conseqüente redução de custos da indústria de eletroeletrônicos. A capacidade de montagem de uma máquina automática SMT pode variar dependendo do fabricante, modelo e custo. Em média, as máquinas utilizadas na indústria são capazes

de montar 20.000 peças/h o que corresponde a uma produção em torno de 60 placas/ h com aproximadamente 300 componentes / placa.

A tendência atual dos produtos eletrônicos, em geral, tem sido a substituição dos componentes '*through-hole*' e discretos (radiais ou axiais) por SMDs. Isso tem contribuído para a redução do tamanho do produto final, dos custos com a matéria prima (componentes) e de fabricação, haja vista que as máquinas automáticas utilizadas para a montagem de SMDs (*Chip Shooters, Pick&Place*, etc.), em geral, são mais ágeis e versáteis do que as máquinas insersoras convencionais.

Além disso, a integração de circuitos em *chips* (CIs – Circuitos Integrados) tem possibilitado a redução da quantidade de peças por placa o que representa um ganho significativo no tempo (e custo) de montagem de placas. A figura 1.3 demonstra a evolução (níveis 1 a 7) da tecnologia de montagem de componentes. Dentre os *chips* mais sofisticados em uso nas indústrias, com níveis de integração superiores aos QFPs (*Quad-Flat Package*), encontram-se: BGA (*Ball Grid Array*), POP (*Package on Package*) e COB (*Chip on Board* / *Wiring Bonding*) correspondentes aos níveis 5, 6 e 7:

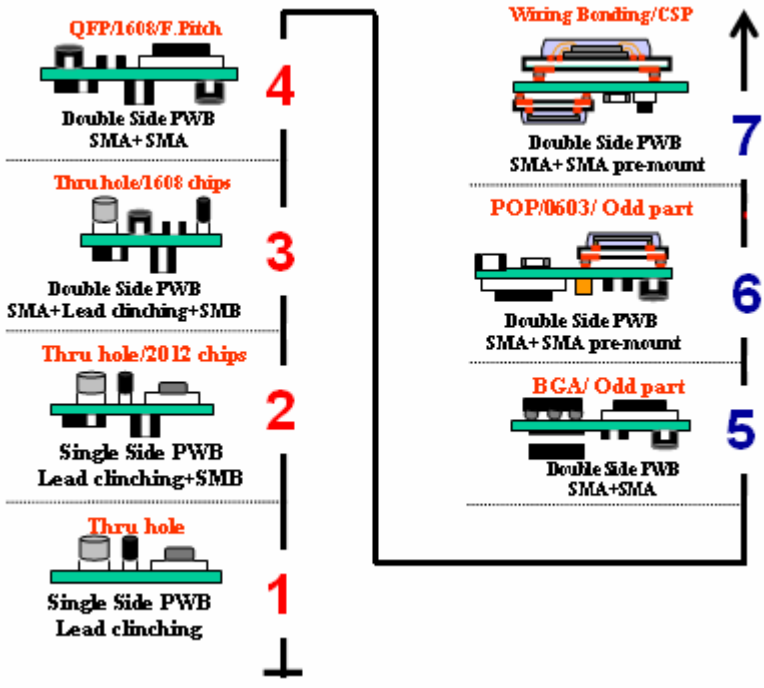


Figura 1.3 – Evolução da Tecnologia de Montagem de Componentes Eletrônicos

A tendência da tecnologia de circuitos integrados é a de reduzir cada vez mais o tamanho do encapsulamento (*downsizing*) e aumentar o número de terminais ou pinos (*increase in pins*), conforme ilustrado na figura 1.4.

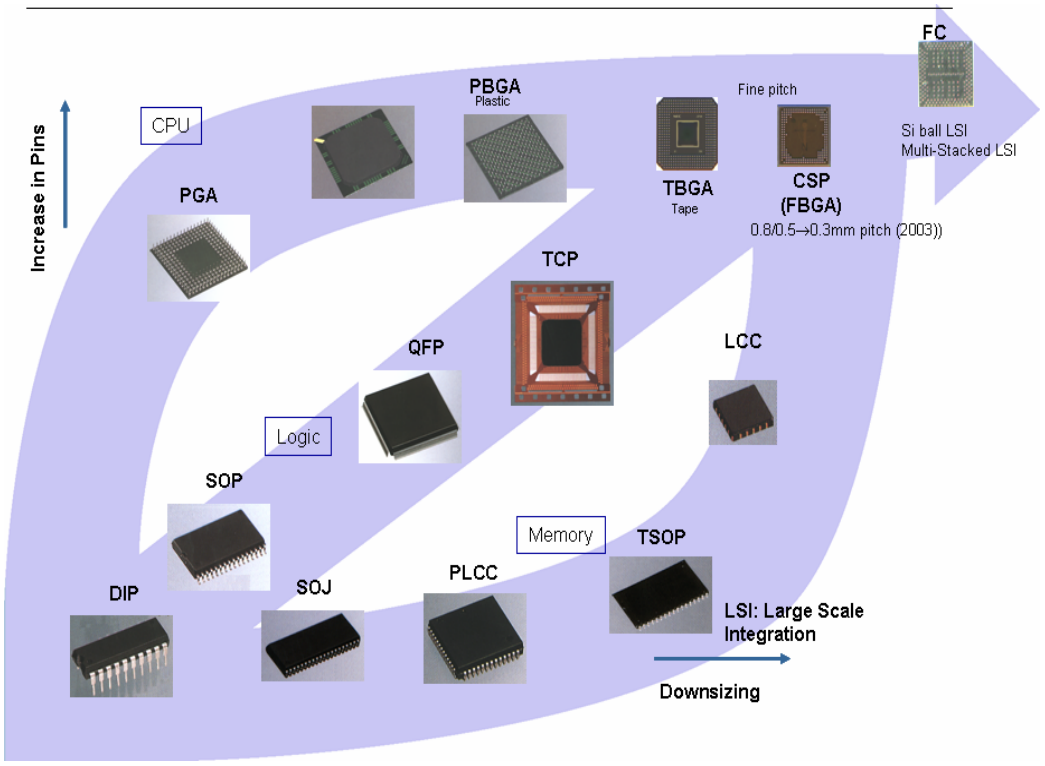


Figura 1.4 – Tendência tecnológica dos Circuitos Integrados (*Downsizing x Increase in Pins*)

A complexidade do problema de **otimização do posicionamento e montagem automática de componentes** está diretamente associada às características de *setup* da máquina que, em geral, compreendem os seguintes subproblemas:

- Balanceamento de máquinas: aplicável quando há mais de uma máquina em série no processo. O balanceamento do tempo de máquina é feito distribuindo-se os componentes entre as diversas estações de trabalho.
- Particionamento *Gantry*: aplicável quando as estações de trabalho são compostas por mais de um conjunto de cabeçotes acoplados em braços (*gantries*) que operam em paralelo.
- **Roteamento ou Seqüenciamento**: compreende a seqüência de operações de *pick-and-place* (busca de componentes nos alimentadores e posicionamento dos mesmos em suas respectivas posições na placa de circuito impresso) executada por cada um dos braços. Tal problema é denominado nesse trabalho de **PAPSP** (*Pick&Place Sequencing Problem*).
- **Arrumação ou configuração dos carretéis com componentes na máquina** (os carretéis são acoplados a dispositivos chamados de alimentadores e estes são encaixados em *slots*. O conjunto de slots adjacentes é chamado de

magazine): este subproblema trata da definição dos *slots* da máquina em que serão alocados os diversos alimentadores com os componentes a serem montados e denomina-se **RAP** (*Reel Assignment Problem*).

- **Alocação de *nozzles* por cabeçote**: subproblema aplicável às máquinas do tipo *multi-head*, em que devem ser estabelecidas as ventosas (*nozzles* ou *tools*) que serão usadas em cada ciclo de *pick&place*, por cada cabeçote, para a captura de componentes nos alimentadores. Chamou-se esse subproblema de **TAP** (*Tool Assignment Problem*).

Dentre esses subproblemas, interessam ao estudo aqui desenvolvido os de **PAPSP**, **RAP** e **TAP**. Para esse último, é apresentada uma abordagem original, que contempla e resolve o caso prático.

1.1. Caracterização do Problema

Nesse trabalho, será considerado para efeito de estudo, modelagem e implementação, um tipo de máquina SMT que, embora complexa, tem se difundido mundialmente pelo baixo custo e alta velocidade de montagem, conhecida como *pick&place Multi-Head Machine* [2], figura 1.5.

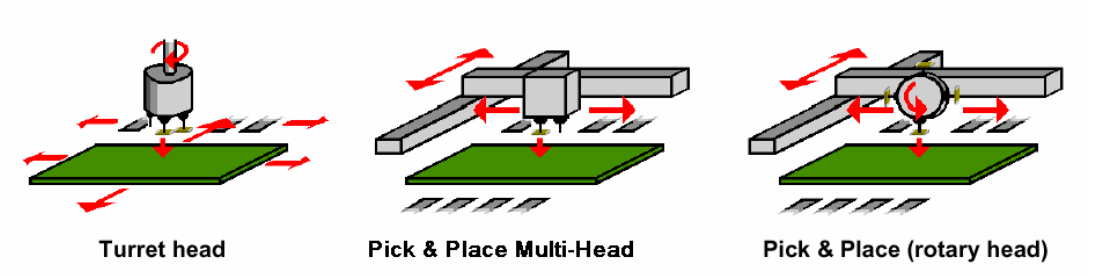


Figura 1.5 – Tipos de Máquina SMT

Na implementação prática do algoritmo proposto nesse trabalho, escolheu-se a máquina *pick&place multi-head* **JUKI modelo KE-2030**, figura 1.6, instalada na empresa multinacional SONY localizada no pólo industrial de Manaus.

Figura 1.6 – Máquina JUKI modelo KE-2030

A figura 1.7, abaixo, ilustra os principais elementos que fazem parte desse sofisticado equipamento:

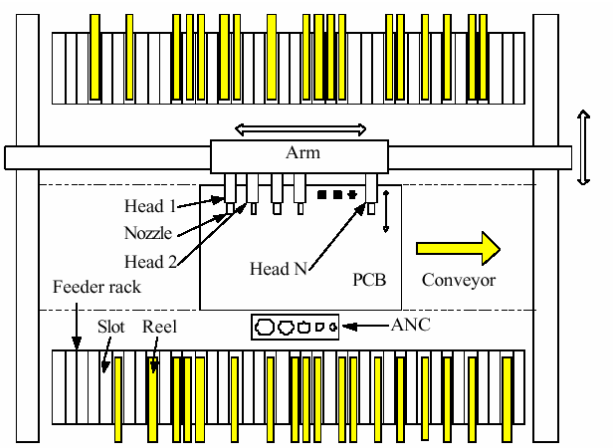


Figura 1.7 – Vista superior (plano) de uma Máquina *Multi-Head* [5]

Seu princípio de operação consiste em um ou mais ciclos de *pick and place* que se iniciam com as múltiplas cabeças fazendo a captura de componentes nos alimentadores. Para reduzir o tempo de captura (*pickup)*, é desejável que os cabeçotes sejam alimentados simultaneamente em slots adjacentes. Em seguida, o braço move-se nas direções X e Y até a posição de montagem de um dos componentes, pois a placa está posicionada em uma mesa fixa. O braço, então, move-se para baixo, na direção Z, e deposita o componente na placa. Isso se repete até que todos os componentes capturados pelas cabeças, no início do ciclo, sejam montados. Para concluir o ciclo, o braço deve retornar aos alimentadores para capturar, assim, novos componentes. O retorno aos

alimentadores poderá ser precedido por uma ida à unidade de troca de *nozzle* (ANC – *Automatic Nozzle Changer*), caso os componentes a serem capturados no ciclo seguinte exijam *nozzles* diferentes daqueles utilizados no ciclo anterior. Vale ressaltar que uma operação de troca de *nozzle* é extremamente lenta quando comparada à velocidade de deslocamento do braço e operação de *pick and place*, devendo, portanto, o número destas ser minimizado.

O problema de otimização pode, portanto, ser descrito nos seguintes termos:

- Deseja-se **minimizar o tempo total de montagem dos componentes em uma placa de circuito impresso**, atuando-se nas variáveis de controle: *pick-up* simultâneo de componentes nos alimentadores, balanceamento da carga de trabalho entre os cabeçotes, distância percorrida pelo braço e trocas de *nozzle*;

Serão feitas, ainda, as seguintes hipóteses para o problema:

- A ordem de acionamento dos cabeçotes é livre e pode ser definida de forma distinta para cada ciclo de *pick and place;*
- Um *slot* pode ser ocupado apenas por um carretel de um determinado tipo de componente;
- Possibilidade de uso simultâneo do mesmo tipo de nozzle em cabeçotes distintos, situação que será denominada no texto de NR (*Nozzle* Repetido);
- A quantidade de componentes em cada carretel é suficiente para a montagem de todo um lote de produção (portanto, não será considerado na modelagem o tempo de recarga de alimentadores: *reloading*).
- Não é permitido carregar nos *slots* mais de um carretel com um determinado componente;

As características da máquina JUKI-KE2030 em estudo são::

- Quantidade de cabeçotes: 4;
- Quantidade de *Gantries*: 2;
- Quantidade de *slots* para alimentadores: 60 (30 frontais e 30 traseiros)
- Tipos de *nozzle*: 4;
- Quantidade de nozzles por tipo: 4;
- Velocidade do braço entre dois pontos quaisquer: constante;

Existem algoritmos industriais (denominados ‘otimizadores’) disponíveis para esse fim, usualmente fornecidos pelo próprio fabricante da máquina. Para o caso específico da máquina JUKI KE-2030 é utilizado o software HLC (*Host Line Computer*).

1.2. Objetivos

- Desenvolver um algoritmo de otimização para a máquina *Multi-Head* que solucione o problema de **minimização do tempo total de montagem dos componentes em uma placa de circuito impresso**, caracterizado na seção anterior.

- Avaliar o desempenho do algoritmo desenvolvido em relação ao algoritmo comercial existente;

- Desenvolver um algoritmo original que tenha um desempenho aceitável e atenda à necessidade prática de utilização de *nozzles* repetidos (NR) nos cabeçotes;

1.3. Organização do Texto

A dissertação está organizada em cinco capítulos. O **Capítulo** 1 apresenta uma introdução e a caracterização do problema de otimização.

No **Capítulo 2,** encontram-se a revisão bibliográfica e a formulação matemática do problema de roteamento da montagem dos componentes, configuração de carretéis e alocação de *nozzles* para os casos *single-head* e *multi-head*.

Os detalhes da implementação do Algoritmo Genético proposto para solucionar o problema de otimização do tempo de montagem de componentes eletrônicos utilizando máquinas tipo *Multi-Head* estão descritos no **Capítulo 3**.

No **Capítulo 4,** são apresentados os resultados obtidos com o algoritmo proposto, comparando-os com os resultados obtidos com o software comercial existente.

No **Capítulo 5**, por fim, encontram-se as considerações finais e propostas para pesquisas futuras.

Ao término do trabalho foi adicionado um **Apêndice** que traz uma revisão da bibliografia sobre algoritmos genéticos, enfatizando o uso desta técnica como solução para problemas de otimização de natureza combinatória.

1.4. Recursos Computacionais

O código desenvolvido foi implementado em linguagem *MATLAB* v.5.3. Os testes foram executados em um microcomputador PC *Intel Pentium IV* de 2.6Ghz, 512 Mb de memória RAM e 256 Kb de cache, operando sob plataforma *Microsoft Windows XP*.

Capítulo 2

Formulação Matemática e Revisão Bibliográfica

O problema físico da montagem de componentes eletrônicos, para o caso da máquina *Multi-Head* com magazine fixo, mesa fixa e conjunto de cabeçotes móvel, pode ser assim descrito:

- **Componentes e alimentadores**: seja uma placa de circuito impresso retangular com *n* posições (com coordenadas XY distintas) a serem ocupadas com seus respectivos componentes eletrônicos. Cada um dos *p* tipos de componentes a serem utilizados está disposto em carretel individual e colocado em máquina por meio de um dispositivo chamado de alimentador. O alimentador pode ser encaixado em um dos *m* slots disponíveis ($m \geq p$) no magazine.
- **Nozzles**: a captura (*pickup*) e posicionamento (*placement*) de um determinado tipo de componente eletrônico é feita por meio de uma ventosa apropriada (*nozzle*) acoplada ao cabeçote. Os fatores que definem o tipo de ventosa a ser utilizado são a área de contato do componente e seu peso. Por exemplo, componentes como circuitos integrados exigem ventosas de maior dimensão, enquanto que componentes tipo 1608 (1,6 x 0,8 mm) exigem ventosas menores. A troca de ventosas é feita na unidade de troca automática (ANC).
- **Montagem**: o ponto de partida do ciclo de montagem de uma placa de circuito impresso é o ANC. A ventosa apropriada à montagem do 1^{o} componente da seqüência é automaticamente instalada no cabeçote. Esse, então, segue em direção ao alimentador em que se encontra o carretel com o

componente a ser montado, executa uma operação de captura (*pickup*) e dirige-se à posição da placa de circuito impresso a ser ocupada pelo componente capturado. Ao atingir a posição-alvo, uma operação de posicionamento (*placement*) é executada, deixando o componente sobre as coordenadas especificadas. Se o próximo componente a ser montado utilizar o mesmo tipo de *nozzle* que está instalado no cabeçote, esse, então, parte em direção ao respectivo alimentador, dando origem a um novo ciclo de *pick & place*. Caso contrário, será necessário, antes, fazer uma visita ao ANC a fim de substituir o *nozzle* por um apropriado e, só então, partir para o novo ciclo.

- **Solução procurada**: busca-se encontrar a melhor seqüência de montagem **(PAPSP)**, configuração dos carretéis nos *slots* do magazine **(RAP)**, alocação de nozzles por cabeçote **(TAP)** que possibilite a montagem de todos os componentes sobre a placa de circuito impresso em **tempo mínimo**.

Os problemas de roteamento (PASPSP) e alocação de carretéis (RAP) são de natureza combinatória. Embora estejam intrinsecamente relacionados, alguns trabalhos publicados apresentam uma formulação desacoplada para efeito de simplificação. Kumar et al. [3] formularam o RAP como um problema de acoplamento de peso mínimo (*Minimum Weight Matching Problem*) e o PAPSP como um TSP. Leipäla et al. [4] modelaram, separadamente, o problema da alocação dos *p* alimentadores nos slots do magazine como um **Problema de Alocação Quadrático** (*QAP – Quadratic Assignment Problem*) e o problema do roteamento ou seqüenciamento da montagem de componentes como o clássico problema do **Caixeiro Viajante (***TSP – Traveling Salesman Problem***),** em que as cidades a serem visitadas são representadas pelas *n* posições na placa. Outros trabalhos [5, 6, 7], no entanto, apresentam uma solução concorrente para os três problemas (PAPSP, RAP e TAP). Na seção 2.4 deste capítulo são descritos maiores detalhes sobre os métodos aproximados utilizados nesses trabalhos e os tipos de máquinas para os quais foram modelados.

Um modelo formal para o problema do roteamento de componentes será apresentado nessa seção, baseando-se no modelo proposto por Burke et al.[7] tendo em vista a interdependência dos problemas de roteamento e alocação dos alimentadores.

Será utilizada a seguinte notação para a descrição do modelo:

‘*n*’ : número total de componentes da placa a serem inseridos

‘*p*’ : número de tipos diferentes de componentes

‘*m*’ : número de *slots* disponíveis ($p \leq m$)

‘*z*’ : número de tipos diferentes de *nozzles* disponíveis

‘*h*’ : número de cabeçotes da máquina

‘*P*’ : conjunto das posições de montagem $\{p_1, p_2, ..., p_n\}$

‘*S*’ : conjunto dos slots disponíveis no magazine $\{s_1, s_2, ..., s_m\}$

‘*C*’ : conjunto dos tipos de componentes $\{c_1, c_2, ..., c_p\}$

‘*N*’ : conjunto dos tipos de nozzles $\{n_1, n_2, ..., n_z\}$

‘δ’ : mapeamento $\delta : C \rightarrow S$ que associa cada tipo diferente de componente a um dos *slots*, não ocupado, do magazine.

‘τ’ : mapeamento $\tau : P \rightarrow N$ que associa um nozzle apropriado a cada uma das posições de montagem.

‘*b*’ : função $b : P \rightarrow C$ que define o tipo de componente a ser usado em cada uma das posições de montagem (na prática, é o próprio B.O.M: *Bill of Materials*).

Seja $\{p_1, p_2, ..., p_n\}$ um conjunto finito de posições de montagem, d_{ij} o custo em termos do tempo gasto para percorrer o trajeto entre p_i e p_j e $\sigma : \{1, 2, ... , n\} \rightarrow \{1, 2, ... , n\}$ uma permutação que estabelece a ordenação da seqüência de montagem. Sendo assim, o problema de roteamento da montagem de componentes corresponde a encontrar σ de forma a:

$$\min_{\sigma} \quad \sum_{i=1}^{n-1} d_{\sigma(i),\sigma(i+1)} + d_{\sigma(n),\sigma(1)} \qquad (2.1)$$

O termo $d_{\sigma(n),\sigma(1)}$ aparece na equação 2.1 pois o processo de montagem automática de componentes é cíclico, ou seja, ao término da montagem do último componente (visita à cidade i = ‘n’) o cabeçote deve retornar à posição inicial (i = 1), similar ao problema de TSP em que: “*um caixeiro viajante quer visitar todo um determinado conjunto de cidades, cada cidade exatamente uma única vez, terminando a*

visita na mesma cidade que começar. Ainda, o custo envolvido no percurso deve ser o menor possível" [8].

A alocação de alimentadores nos *slots* do magazine é de fundamental importância para o problema, uma vez que influencia diretamente no tempo gasto entre a montagem de dois componentes consecutivos, d_{ij}, e consequentemente na seqüência ótima.

2.1. Caso *Single-Head* (SH)

Seja uma máquina com cabeçote único (*Single Head*) com um magazine de *m slots* $\{s_1, s_2, ..., s_m\}$ disponíveis para alocação dos alimentadores e uma placa a ser montada com os componentes $\{c_1, c_2, ..., c_p\}$ nas posições $\{p_1, p_2, ..., p_n\}$. Define-se o mapeamento $\delta: \{c_1, c_2, ..., c_p\} \rightarrow \{s_1, s_2, ..., s_m\}$ que correlaciona os componentes e os *slots* do magazine em que estão alocados seus respectivos carretéis ou alimentadores.

Define-se $d^{\delta}_{i,j}$ como o tempo gasto para percorrer o trajeto entre p_i e p_j para uma determinada configuração δ dos carretéis de componentes nos *slots* do magazine.

O problema de roteamento da montagem de componentes, para uma máquina com cabeçote único, pode então ser escrito como um problema de otimização em dois níveis, buscando-se encontrar σ e δ a fim de:

$$\underset{\sigma,\delta}{\text{minimizar}} \quad \sum_{i=1}^{n-1} d^{\delta}_{\sigma(i),\sigma(i+1)} + d^{\delta}_{\sigma(n),\sigma(1)} \tag{2.2}$$

Na equação 2.2, em comparação com a equação 2.1, as distâncias entre as cidades, para o problema do caixeiro viajante, dependem da alocação dos carretéis de componentes nos *slots* do magazine.

Um aspecto importante que, também, necessita fazer parte do modelo matemático é a inclusão da possibilidade de troca de *nozzles* no decorrer da montagem. Existem *nozzles* apropriados aos diversos tipos de componentes em função da sua área de contato e peso conforme descrito anteriormente.

O problema de roteamento está, portanto, relacionado à questão da troca de *nozzles* pois o tempo entre a montagem de dois componentes consecutivos depende, além da localização dos alimentadores no magazine, da necessidade ou não de uma visita ao ANC. Seja, então, $\{n_1, n_2, ..., n_z\}$ o conjunto de *nozzles* disponíveis na máquina e $\tau : \{p_1, p_2, ..., p_n\} \rightarrow \{n_1, n_2, ..., n_z\}$ uma função que associa cada uma das posições de montagem p_i a um *nozzle* n_j apropriado. Define-se $d_{i,j}^{\delta,\tau}$ como o tempo necessário para o cabeçote montar um componente na posição p_i, utilizando um *nozzle* $\tau(p_i)$, efetuar uma troca de *nozzle* se $\tau(p_i) \neq \tau(p_j)$, capturar um componente no carretel que está do *slot* $\delta(b(p_j))$, e montá-lo na posição p_j.

Esse problema pode, então, ser escrito como um problema de otimização em três níveis, buscando-se encontrar σ, δ e τ a fim de:

$$\underset{\sigma,\delta,\tau}{\text{minimizar}} \quad \sum_{i=1}^{n-1} d_{\sigma(i),\sigma(i+1)}^{\delta,\tau} + d_{\sigma(n),\sigma(1)}^{\delta,\tau} \tag{2.3}$$

A equação 2.3 estabelece que o tempo total de montagem dos componentes eletrônicos na placa de circuito impresso depende não só de sua seqüência σ, mas também da configuração δ dos carretéis nos *slots* do magazine e da alocação τ de *nozzles* apropriados aos componentes.

Para o caso *single head* é trivial que τ e σ possuem uma relação muito próxima, uma vez que, para minimizar a troca de *nozzles*, τ irá ordenar o trajeto $p_{\sigma(1)}, p_{\sigma(2)}, ..., p_{\sigma(n),} p_{\sigma(1)}$ em segmentos correspondentes a *nozzles* diferentes.

Exemplo:

Sejam os grupos de componentes que utilizam o mesmo tipo de nozzle:

$$\{i : \tau(p_i) = n_1\} = \{1, 3, 5\}$$

$$\{i : \tau(p_i) = n_2\} = \{2, 4, 6\}$$

As possíveis seqüências de montagem σ que minimizam as trocas entre n_1 e n_2 podem ser escritas como:

σ = $\{\{p_1, p_3, p_5\}, \{p_2, p_4, p_6\}\}$ e todas as permutações possíveis entre as componentes que fazem parte do mesmo segmento ou

σ = $\{\{p_2, p_4, p_6\}, \{p_1, p_3, p_5\}\}$ e todas as permutações possíveis entre as componentes que fazem parte do mesmo segmento, perfazendo um total de 3! vezes 3! = 36 possibilidades nesta ordem, + 36 na outra = 72 possibilidades.

2.2. Caso *Multi-Head* (MH): modelo utilizando o conceito de Hiperaresta & Hipergrafo

Será apresentado, nessa seção, um modelo para a máquina com múltiplos cabeçotes (*Multi-Head*), baseado em Burke et al. [9, 10] e Keuthen [11], como uma extensão do modelo *Single-Head* da seção anterior. Para esse tipo de máquina existem, também, os três subproblemas descritos: RAP, PAPSP, TAP.

RAP: os diversos alimentadores, com tipos diferentes de componentes, devem ser alocados, seguindo um dos possíveis mapeamentos δ, em *slots* distintos do magazine;

PAPSP: deve ser determinada uma seqüência σ ótima de inserção dos componentes que resulte no mínimo tempo total de montagem de uma placa.

TAP: *nozzles* apropriados devem ser associados, pela função τ, aos diversos tipos de componentes;

Alguns trabalhos disponíveis na literatura tratam esses sub-problemas separadamente e apresentam soluções independentes para cada um. Burke et al. [9] propõe uma abordagem de otimização em múltiplos níveis, descrita a seguir, tendo em vista a forte inter-relação desses três subproblemas.

A idéia básica desse modelo [9] é considerar o problema de seqüenciamento da montagem de componentes em uma máquina *multi-head* como uma generalização do problema do caixeiro viajante, aqui denominado de problema do Hipergrafo com custo mínimo.

Define-se um ciclo de captura múltipla de componentes e a correspondente seqüência de montagem desses sobre a placa como sendo uma seqüência dirigida ou Hiperaresta $H_i = \left(p_1^i, p_2^i, ..., p_{l_i}^i \right)$, onde $p_j^i \in P$ e $l_i = |H_i| \leq h+1$. Define-se, ainda, $|H_i|$

como sendo a cardinalidade da Hiperaresta H_i, ou seja, o número de elementos p_j^i compreendidos em H_i.

Uma Hiperaresta H_i corresponde a uma sub-seqüência que se inicia com todos os cabeçotes vazios na posição p_1^i, a partir de onde o braço dirige-se em direção aos alimentadores do magazine onde estão localizados os componentes a serem montados nas posições $p_2^i,..., p_{l_i}^i$. Em seguida, o braço retorna em direção a placa posicionando os componentes capturados na seqüência $p_2^i,..., p_{l_i}^i$, terminando com os cabeçotes vazios na posição $p_{l_i}^i$.

Um Hipergrafo $G = (H_1, H_2, ..., H_k)$ é definido como uma seqüência dirigida de Hiperarestas adjacentes e que satisfazem às condições abaixo:

(1): $2 \le |H_i| \le h+1, i = 1,...,k.$

Assegura que cada Hiperaresta consiste de pelo menos um componente e não mais do que o número de cabeçotes *h* disponíveis.

(2): $p_{l_i}^i = p_1^{i+1}, i = 1,...,k-1$ e $p_{l_k}^k = p_1^1$

Assegura que o término de uma Hiperaresta coincide com o início da próxima.

(3): $|H_i \cap H_{i+1}| = 1, i = 1,...,k-1$

$|H_k \cap H_1| = 1$

$H_i \cap H_j = \varnothing, i, j = 1,...,k, |i-j| > 1$ e $\{i, j\} \neq \{1, k\}$

Assegura que uma posição de montagem não será visitada mais de uma vez, exceto a primeira (índice = 1).

(4): $\bigcup_{j=1}^{k} H_j = \{p_1, p_2, ..., p_n\}$

Assegura que todas as posições de montagem serão visitadas.

A fim de minimizar o tempo total de montagem dos componentes na máquina deve-se minimizar os tempos associados aos ciclos de *pick & place* de todas as seqüências (Hiperarestas) que constituem o Hipergrafo. A fim de avaliar a qualidade de uma seqüência, faz-se necessário definir tempos associados ao deslocamento entre as cidades (posições de montagem). Seja $D^{\sigma,\tau}(H)$ o tempo necessário para percorrer a

seqüência de *pick & place* da Hiperaresta *H* sujeitos para uma alocação σ de carretéis nos *slots* e uma alocação τ de *nozzles* aos componentes a serem montados na placa.

Dessa forma, é possível representar os subproblemas RAP, PAPSPS e TAP como um problema de otimização em 3 níveis, sujeito às restrições (1)-(4) acima, dado por:

$$\min_{S,\sigma,\tau} \sum_{i=1}^{k} D^{\sigma,\tau}(H_i)$$

2.3. Caso *Multi-Head* (MH): formulação como problema de Programação Matemática

Nessa seção, será apresentada uma formulação proposta por Ayob et al. [12] do problema de otimização de uma função de tempo de montagem de componentes eletrônicos em uma máquina automática tipo *Multi-Head* como um problema de Programação Matemática. Os principais elementos dessa máquina *Multi-Head*, ilustrada na figura 1.7, que influem na modelagem, são:

- Braço equipado com "*h*" cabeçotes
- Mesa fixa de apoio para a placa de circuito impresso
- Magazine fixo e duplo (frente e trás) para disposição dos alimentadores
- Braço móvel com movimentação simultânea nas direções X e Y

A função constitui-se na minimização do tempo total de montagem como conseqüência da minimização da distância percorrida pelo braço robótico, para concluir todos os **ciclos** (denomina-se **ciclo** como sendo uma operação de captura e montagem de componentes ou, simplesmente, ciclo de *pick&place*).

Será utilizada a seguinte notação para a construção do modelo:

'CT' : tempo para a montagem de todos os componentes de uma placa
'n' : número total de componentes a serem montados na placa
'p' : número de tipos de componentes (considera-se que cada alimentador possui uma quantidade infinita de componentes)
'h' : número de cabeçotes

‘*nc*’ : número total de ciclos de *pick&place*.

‘*m*’ : número total de *slots* para alojamento de alimentadores ($p \leq m$)

‘$c(j,k)_{x,y}$’ : coordenada X, Y na PCI em que o componente será inserido na *k-ésima* seqüência de montagem

‘*V*’ : velocidade de deslocamento (constante) do braço robótico

‘λ’ : tempo de captura (*pickup*) de um componente

‘θ’ : tempo para inserção (*placement*) de um componente

‘*r*’ : *r-ésimo* slot, onde r $\in$ { 0, 1, 2, ... , (*m*-1)};

‘*i*’ : *i-ésimo* tipo de componente, onde i $\in$ {1, 2, ... , *p*};

‘*k*’ : *k-ésima* seqüência de *pickup* em um ciclo, onde k $\in$ {1, 2, ... , *h*};

‘*f*’ : *f-ésima* seqüência de montagem em um ciclo, onde f $\in$ {1, 2, ..., *h*};

‘*j*’ : *j-ésimo* ciclo de *pick&place*, onde j $\in$ {1, 2, ... , *nc*};

‘$I(j,k)$’ : tempo gasto pelo braço para percorrer a distância entre o alimentador e um ponto da placa e inserir um componente na *k-ésima* seqüência de montagem do *j-ésimo* ciclo.

‘$P(j,k)$’ : tempo gasto pelo braço para percorrer a distância entre um ponto da placa e o alimentador e capturar um componente na *k-ésima* seqüência de *pickup* do *j-ésimo* ciclo.

‘$\Phi_i(j,k)$’ : cabeçote utilizado para capturar o *i-ésimo* componente na *k-ésima* seqüência de *pickup* do *j-ésimo* ciclo.

‘$\Omega_i(j,f)$’ : cabeçote utilizado para inserir o *i-ésimo* componente na *f-ésima* seqüência de montagem do *j-ésimo* ciclo.

‘$S(r)$’ : distância do *r-ésimo slot* à origem do magazine, S(0), em que S(0) = 0 e $r \geq p$.

‘$R(j,k)$’ : distância do *slot* para a k-ésima seqüência de *pickup* do *j-ésimo* ciclo.

‘$\hat{d}(j,k)$’ : max{ $|d(j,k)_x|$, $|d(j,k)_y|$ }, onde *x* e *y* são as distâncias X, Y percorridas pelo braço robótico para inserir um componente na *k-ésima* seqüência de montagem do *j-ésimo* ciclo, em que as distâncias calculadas são de Chebychev (estabelecidas pelo valor máximo das distâncias percorridas nas direções X ou Y, uma vez que o braço move-se em ambas as direções concorrentemente).

'$\hat{m}(j,k)$' : max{ $|m(j,k)_x|$, $|m(j,k)_y|$ }, onde *x* e *y* são as distâncias X, Y percorridas pelo braço robótico para capturar um componente na *k-ésima* seqüência de *pickup* do *j-ésimo* ciclo, em que as distâncias calculadas são de Chebychev.

'F' : intervalo (*gap*) entre dois *slots* adjacentes.

'L' : intervalo (*gap*) entre dois cabeçotes adjacentes.

Problema de Otimização:

Minimizar

$$CT = \sum_{j=1}^{nc} \left[\sum_{k=1}^{h} P(j,k) + \sum_{k=1}^{h} I(j,k) \right] \tag{2.4}$$

sujeito à:

$\sum_{i=1}^{p} g_{ir} \le 1, \forall r;$ cada alimentador pode conter apenas um tipo de componente (2.5)

$\sum_{r=0}^{m-1} g_{ir} = 1, \forall i;$ assumindo que não há duplicidade de alimentadores (2.6)

$$g_{ir} = \begin{cases} 1: \text{ se o componente tipo ' i ' estiver associado ao slot ' r '} \\ 0: \text{ caso contrário} \end{cases} \tag{2.7}$$

$\Phi_i(j,k) \in \{0,1,2,...(h-1)\};\ \varphi(j,k) \neq \varphi(j,l)$ se $k \neq l;\ j = 1,2,..,nc; k = 1,2,..,h;$ (2.8)

$\Omega_i(j,f) \in \{0,1,2,...(h-1)\};\ \omega(j,f) \neq \omega(j,l)$ se $f \neq l;\ j = 1,2,..,nc;\ f = 1,2,..,h;$ (2.9)

$\Phi_i(j,k) = \Omega_i(j,f)$ (2.10)

$$z(j,k) = \begin{cases} 1: \text{ se há um cabeçote escalado para capturar ou inserir um componente} \\ \quad \text{na k-ésima seqüência do j-ésimo sub-trajeto} \\ 0: \text{ caso contrário.} \end{cases} \tag{2.11}$$

$\sum_{k=1}^{h} z(k,j) \le h$ (2.12)

$$b_r(j,k) = \begin{cases} 1: \text{ se há um componente a ser capturado no slot "r" na} \\ \quad \text{k-ésima seqüência de pickup do j-ésimo sub-trajeto.} \\ 0: \text{ caso contrário.} \end{cases} \tag{2.13}$$

$\sum_{r=0}^{m-1} b_r(j,k) \le 1, \forall j, \forall k;$ somente 1 comp. pode ser capturado em cada pickup. (2.14)

onde:

$$nc = \begin{cases} n/h, \text{ se } n \text{ MOD } h = 0; \\ 1 + n/h, \text{ se } n \text{ MOD } h \neq 0; \end{cases} \tag{2.15}$$

$$P(j,k) = \left(\frac{\hat{m}(j,k)}{V} + \lambda \right) * z(j,k); \tag{2.16}$$

$$I(j,k) = \left(\frac{\hat{d}(j,k)}{V} + \theta \right) * z(j,k); \tag{2.17}$$

$$m(j,k)_x = \begin{cases} R(j,k) - \varphi(j,k) * L, \text{ se } j = k = 1; \\ \left[R(j,k) - \varphi(j,k) * L \right] - \left[c(j-1,h)_x - \omega(j-1,h) * L \right], \text{ se } k = 1,\ j > 1; \\ \left[R(j,k) - R(j,k-1) \right] - \left[\left(\varphi(j,k) - \varphi(j,k-1) \right) * L \right], \text{ se } k > 1; \end{cases} \tag{2.18}$$

$$d(j,f)_x = \begin{cases} \left[c(j,f)_x - R(j,f) \right] - \left[\left(\omega(j,f) - \varphi(j,h) \right) * L \right], \text{ se } f = 1; \\ \left[c(j,f)_x - c(j,f-1)_x \right] - \left[\left(\omega(j,f) - \omega(j,f-1) \right) * L \right], \text{ se } f > 1; \end{cases} \tag{2.19}$$

$$m(j,k)_y = \begin{cases} 0 - \left[c(j-1,h) \right], \text{ se } f = 1; \\ 0 \text{ caso contrário.} \end{cases} \tag{2.20}$$

$$d(j,f)_y = \begin{cases} c(j,f)_y, \text{ se } f = 1; \\ \left[c(j,f)_y - c(j,f-1)_y \right], \text{ se } f > 1; \end{cases} \tag{2.21}$$

$$S(r) = r * F \tag{2.22}$$

$$R(j,k) = \sum_{r=0}^{m-1} \left[S(r) * b_r(j,k) \right] \tag{2.23}$$

O tempo total de montagem, CT, é uma função do deslocamento do braço dividido pela sua velocidade, somado aos tempos de *pickup* (λ) e *placement* (θ).

Um trajeto completo da máquina consiste de '*nc*' ciclos de *pick&place* e cada um desses ciclos possui, no máximo, '*h*' pares de pontos de captura e montagem (equação 2.4).

Observa-se, ainda, que não foi levada em consideração nessa formulação a possibilidade de haver troca de nozzle em um determinado cabeçote. Mesmo com essas restrições, é notória a complexidade da formulação como um problema de Programação Matemática e, consequentemente, a dificuldade de resolver o problema dessa forma. Um modelo alternativo proposto por Lee et al. [5], que contempla a troca de nozzles em sua formulação, está descrito no Capítulo 3.

Com vista a solucionar as equações formuladas para o problema de otimização proposto nesse trabalho, a seção a seguir apresenta alguns métodos aproximados de otimização combinatória que simplificam a modelagem e resolução quando comparados aos métodos exatos existentes na literatura.

2.4. Soluções para o problema utilizando Métodos Aproximados

Muitos problemas práticos são modelados da seguinte forma: "Dado um conjunto S de variáveis discretas s, denominadas soluções, e uma função objetivo $f : S \to R$, que associa cada solução $s \in S$ a um valor real $f(s)$, encontre a solução $s^* \in S$, dita ótima, para a qual $f(s)$ é mínima".

Tais problemas são enquadrados como problemas de otimização combinatória. Dentre os mais conhecidos dessa categoria encontra-se o problema do Caixeiro Viajante (*TSP – Traveling Salesman Problem*).

O *TSP* é descrito por um conjunto de n cidades e uma matriz de distâncias (ou custos) entre elas, tendo o seguinte objetivo: o caixeiro viajante deve sair de uma cidade chamada origem, visitar cada uma das restantes $n-1$ cidades apenas uma única vez e retornar à cidade origem percorrendo a menor distância possível. Em outras palavras, deve ser encontrada uma rota fechada (ciclo hamiltoniano) de comprimento mínimo que passe exatamente uma única vez por cada cidade.

O *TSP* torna-se complexo a medida em que o número de cidades torna-se alto. Assumindo que as distâncias entre duas cidades i e j sejam simétricas, isto é, $d_{ij} = d_{ji}$, o número total de rotas possíveis é: $(n-1)!/2$. Para n elevado, enumerar todas as possíveis rotas torna-se computacionalmente intratável.

Assim, em problemas dessa natureza, o uso de métodos exatos se torna bastante restrito. Por outro lado, na prática, em geral, é suficiente encontrar uma solução sub-ótima para o problema, ao invés de uma solução ótima. Este é o motivo pelo qual os pesquisadores têm lançado mão de heurísticas para resolver problemas desse nível de complexidade. Define-se **heurística** como uma técnica que procura boas soluções (próximas da otimalidade) a um custo computacionalmente razoável sem, no entanto, estar capacitada a garantir a otimalidade, bem como garantir o quão próximo uma solução está da solução ótima.

A partir de 1980 intensificaram-se os estudos no sentido de desenvolver procedimentos heurísticos, com uma certa estrutura teórica e de caráter mais geral, sem o entanto prejudicar a sua principal característica: a flexibilidade. Essa meta tornou-se mais realista a partir da reunião dos conceitos de Otimização e Inteligência Artificial, viabilizando a construção das chamadas melhores estratégias ou dos métodos inteligentemente flexíveis, comumente conhecidos como **metaheurísticas**.

Metaheurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual precisa ser modelada para cada problema específico.

Dentre os procedimentos que se enquadraram como metaheurísticas, surgidos ao longo das últimas décadas, destacam-se: **Algoritmos Genéticos** (AGs), Redes Neurais, *Simulated Annealing* (SA), Busca Tabu (BT), GRASP (*Greedy Randomized Adaptive Search Procedure*) , VNS (*Variable Neighborhood Search*), etc.

As duas primeiras metaheurísticas fundamentam-se em analogias com processos naturais, enquanto a SA explora uma possível analogia com a Termodinâmica e a BT faz uso de uma memória flexível para tornar o processo de busca mais eficaz.

A figura 2.1 apresenta diversos métodos existentes na literatura voltados à solução de problemas de otimização combinatória, com especial destaque aos Algoritmos Genéticos (AGs) tendo em vista a sua escolha para a realização deste trabalho.

Os modelos matemáticos, apresentados neste capítulo, tratam o problema de otimização combinatória da montagem de componentes eletrônicos em uma placa de circuito impresso, utilizando uma máquina *pick & place* tipo *Multi-Head*, como um único problema de otimização em três níveis:

- Configuração dos carretéis nos *slots* da máquina (RAP – *Reel Assignment Problem*);

- Roteamento ou seqüenciamento de montagem dos componentes (PAPSP – *Pick & Place problem*);
- Alocação de *nozzles* nos cabeçotes (TAP – *Tool Assignment Problem*)

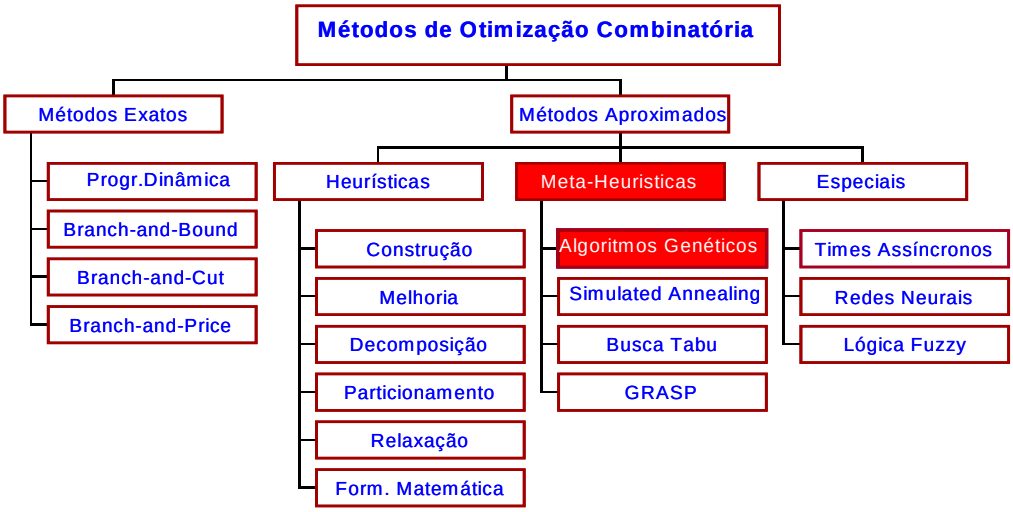


Figura 2.1 – Métodos de Otimização Combinatória

Vários métodos aproximados de solução foram propostos nos trabalhos encontrados na literatura para diferentes tipos de máquinas, representados em uma forma sintetizada na figura 2.2, destacando-se: BALL et al. [13] utilizaram heurísticas para separar os problemas de RAP e PAPSP em dois problemas desacoplados e propuseram uma solução individual para cada um, considerando o uso de uma máquina insersora (*sequential inserter*) de componentes '*through-hole*'.

Raros trabalhos foram desenvolvidos propondo uma solução para o problema do TAP, além dos problemas de RAP e PAPSP. Entre eles, encontram-se: Lee et al. [5] utilizaram algoritmos genéticos com cromossomos (*links)* parciais para modelar de forma concorrente os problemas de RAP e PAPSP em uma máquina tipo *pick&place* com múltiplos cabeçotes (MH: *Multi-Head*). O TAP foi tratado separadamente utilizando um AG simples com restrição de não repetir o mesmo nozzle em mais de um cabeçote. Tirpak et al. [6] utilizaram um método de *Simulated Annealing* Adaptativo para máquinas MH do tipo *Turret, Rotary ou Revolver-Head.* Para este mesmo tipo de máquina, Ho [14] propôs o método de Algoritmos Genéticos Híbridos (HGA) para tratar de forma simultânea os problemas de RAP e PAPSP.

Jeevan et al. [15] também fizeram uso de AGs para solucionar os problemas de RAP e PAPSP como um TSP, em uma máquina MH tipo *pick&place*, adicionando na modelagem do TSP o fator de troca de nozzle (*tool change factor*), sem no entanto abordar a possibilidade de uso de nozzles repetidos e mais de um cabeçote. Burke et al. [9] modelaram os três problemas (RAP, PAPSP e TAP) como um único problema de otimização em três níveis, no entanto, para a solução propuseram o uso de heurística (construção) e metaheurística (VNS – *Variable Neighborhood Search*) voltados aos problemas de RAP e PAPSP, e negligenciaram o TAP assumindo que cada cabeçote da máquina MH tipo *pick&place* é equipado com um nozzle universal capaz de capturar e posicionar componentes de quaisquer tipos.

Tipo de Máquina	Sequential Inserter (*Robot-Arm*)		Turret or Revolver Head (*chip shooter*)		Pick & Place	
Modelo	PANASERT RH (Model NM-8202A)		FUJI NP-132		Não especificado	
Tipo de Componente	Through-Hole		SMD (chip)		SMD (chip)	
Multiplicidade de Cabeçotes	Single-Head		Multi-Head		Multi-Head	
Bibliografia (Autor) / Método de Resolução	Leipala[4]	Heurísticas (Farthest Insertion, Pairwise Exchange, MST)	Tirpak[6]	Adaptive Simulated Annealing	Lee[5]	Algoritmos Genéticos
	Ball [6]	Heurísticas (RAP e PAPSP desacoplados)	Ho[14]	Hybrid Genetic Algorithm	Jeevan [15]	Algoritmos Genéticos
					Burke [7], [9] e [10]	Heurísticas (Construction, Local Search, VNS)

Figura 2.2 – Características de Máquinas e trabalhos encontrados na literatura

Em vista da simplicidade de modelagem de um problema de otimização combinatória e do sucesso obtido em resolver problemas de otimização complexos, escolheu-se, nesse trabalho, a técnica dos **Algoritmos Genéticos** (AG) para modelar e resolver os problemas de RAP, PAPSP e TAP associados à montagem de placa de circuito impresso em máquinas *pick&place* do tipo *Multi-Head*.

Capítulo 3

Algoritmo Genético Proposto

Tendo sido formulado o problema de otimização da montagem automática de componentes eletrônicos sobre uma placa de circuito impresso, utilizando máquinas tipo *Multi-Head*, que engloba os sub-problemas de roteamento, alocação de componentes nos alimentadores e associação de *nozzles* aos cabeçotes, será descrita, a seguir, a implementação do AG, desenvolvido nesse trabalho, para solucioná-lo.

O AG implementado foi baseado na solução proposta por [5], um dos raros trabalhos encontrados na literatura sobre esse tipo de máquina, com uma inovadora e significativa modificação:

- A modelagem e resolução do caso MH-NR (*Multi-Head* com *Nozzle* Repetido). Lee et al. [5], restringe o problema a um único nozzle de cada tipo. Isso impõe a condição de cada cabeçote sempre utilizar um *nozzle* distinto dos demais, o que, embora simplifique bastante o algoritmo, não se adequa ao caso prático. Isso porque, em problemas reais, há quase sempre um desbalanceamento entre as quantidades de componentes atribuídos a cada tipo de *nozzle*, resultando em soluções ineficientes (devido ao conseqüente uso desbalanceado dos cabeçotes). Nesse trabalho, foi considerada a possibilidade do uso do mesmo tipo de nozzle em mais de um cabeçote, simultaneamente, um caso a que se denominou de MH-NR.

Inicialmente, será descrita a técnica utilizada para resolver o caso *single-head* (cabeçote único) e, em seguida, uma extensão do conceito aplicada ao caso mais complexo, *Multi-Head* (cabeçote múltiplo). Para ambos os casos, definiu-se os parâmetros físicos da máquina conforme abaixo:

‘*n*’ : número de peças a serem montados por placa (posições de montagem)

‘*nc*’ : número de ciclos de *pick&place* por placa

‘*p*’ : quantidade de tipos de peças ou componentes

‘*m*’ : quantidade de *slots*

‘*H*’ : quantidades de cabeçotes

‘*C*’ : conjunto das posições de montagem dos componentes

‘*S*’ : conjunto dos *slots*

‘*b(c)*’ : tipo de componente montado na posição $c \in C$

‘$T(p_1, p_2)$’ : tempo de deslocamento entre dois pontos, p_1 e p_2.

‘*Tanc*’ : tempo de troca de *nozzle*

‘h_i’ : *i-ésimo* cabeçote

‘N_i’ : cj. dos nozzles utilizados nos componentes montados nas posições $c_i \in C$

3.1. Caso *Single-Head* (SH)

Haja vista a interdependência dos problemas de roteamento e alocação de componentes nos alimentadores, é natural que ambos sejam resolvidos de forma concorrente. Para isso, será adotada a representação dos cromossomos na forma de *links* parciais (*Partial Links*) [5].

3.1.1. Formação dos cromossomos

A estrutura de cromossomo desenvolvida consiste em dois *links* parciais, A e B, que representam, respectivamente, a alocação de componentes nos alimentadores e o roteamento. A representação utilizada foi a do tipo **Permutação** [16] (*Permutation Representation*), também conhecida como **Rota** [17] (*Path Representation*) ou **Lista Ordenada** (*Ordered List*), por ser a forma mais natural de representar a seqüência das cidades a serem visitadas pelo Caixeiro Viajante, nesse caso, o braço robótico.

O *link* A é, na prática, chamado de **mapa de alimentação** (Tabela 3.1), cuja primeira coluna, "Z (feeder)", está diretamente representada no cromossomo. É definido como $\tilde{l} = [l(1), l(2), ..., l(m)]$, em que $l(i) \in S$, $l(i) \neq l(j)$ se $i \neq j$, $\forall i, j \in \{1, 2, ..., m\}$. $l(i)$ é um gene que representa o número do slot em que o componente tipo i está alocado (figura 3.1). Como a quantidade de tipos de componentes (p) é inferior ao número de slots (m), sobram $m - p$ genes sem uso nesse *link*.

Tabela 3.1 - Mapa de Alimentação

Z (Feeder)	CÓDIGO	CompType	VALOR	P	QTY	REFERÊNCIA
3	1-162-964-91	1	CAP. CHIP 1000PF 1608		2	a1, a2
5	1-162-960-91	2	CAP. CHIP 220PF 1608		3	b1, b2, b3
6	1-216-845-91	3	RES. CHIP 100K 1608		2	c1, c2
...	1-216-841-91	4	RES. CHIP 47K 1608		2	d1, d2
	1-164-156-91	5	CAP. CHIP 0,1MF 1608		3	e1, e2, e3
	1-216-134-91	6	RES. CHIP 2,2 (3216)		1	f1
	1-216-853-91	7	RES. CHIP 470K 1608		1	g1
	1-216-857-91	8	RES. CHIP 1M 1608		1	h1
	1-216-809-91	9	RES. CHIP 100 1608		1	i1
	1-216-825-91	10	RES. CHIP 2,2K		3	j1, j2, j3
	1-216-837-91	11	RES. CHIP 22K 1608		1	k1
	1-216-829-91	12	RES. CHIP 4,7K 1608		1	l1

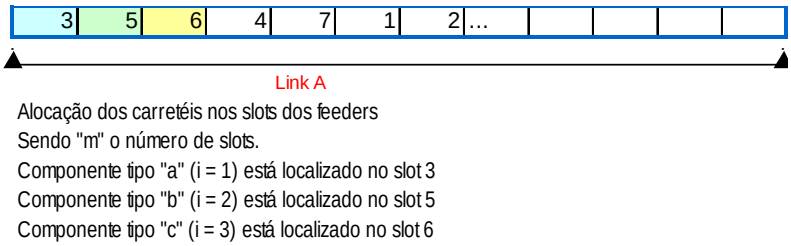


Figura 3.1 - Estutura do LinkA

O *link* B é, por sua vez, na prática, a própria **seqüência (ou programa) de montagem** (figura 3.2). É definido como $\tilde{l^*} = [l^*(1), l^*(2), ..., l^*(n)]$, em que $l^* \in C$, $l^*(i) \neq l^*(j)$, se $i \neq j, \forall i, j \in \{1, 2, ..., n\}$. $l^*(i)$ representa o *i-ésimo* componente a ser montado em seqüência (Tabela 3.2).

Tabela 3.2 – Exemplo de um programa de montagem

No# (Sequencia)	CompID	X	Y	Angle	Compo	CompType	name	Plac	Try	Skip	Lyr	Sta	GX4NozzleNo
9	d1	8	5	0.00	1-216-841-91	4	Plac	No	No	4	-		1
8	c2	8	4	0.00	1-216-845-91	3	Plac	No	No	4	-		1
1	a1	10	4	180.00	1-162-964-91	1	Plac	No	No	4	-		1
3	d2	12	5	0.00	1-216-841-91	4	Plac	No	No	4	-		1
...	c1	12	3	0.00	1-216-845-91	3	Plac	No	No	4	-		1
	a2	10	5	0.00	1-162-964-91	1	Plac	No	No	4	-		1
	h1	20	5	0.00	1-216-857-91	8	Plac	No	No	4	-		2
	k1	22	6	0.00	1-216-837-91	11	Plac	No	No	4	-		2
	f1	18	3	0.00	1-216-134-91	6	Plac	No	No	4	-		2
	j3	20	4	90.00	1-216-825-91	10	Plac	No	No	4	-		3
	e1	18	6	90.00	1-164-156-91	5	Plac	No	No	4	-		3
	b3	20	6	90.00	1-162-960-91	2	Plac	No	No	4	-		3
	j1	16	4	90.00	1-216-825-91	10	Plac	No	No	4	-		3
	e2	18	5	90.00	1-164-156-91	5	Plac	No	No	4	-		3
	b2	16	2	90.00	1-162-960-91	2	Plac	No	No	4	-		3
	l1	12	4	90.00	1-216-829-91	12	Plac	No	No	4	-		3
	i1	13	3	90.00	1-216-809-91	9	Plac	No	No	4	-		3
	g1	10	3	90.00	1-216-853-91	7	Plac	No	No	4	-		3
	j2	24	5	0.00	1-216-825-91	10	Plac	No	No	4	-		3
	e3	24	4	90.00	1-164-156-91	5	Plac	No	No	4	-		3
	b1	24	6	90.00	1-162-960-91	2	Plac	No	No	4	-		3

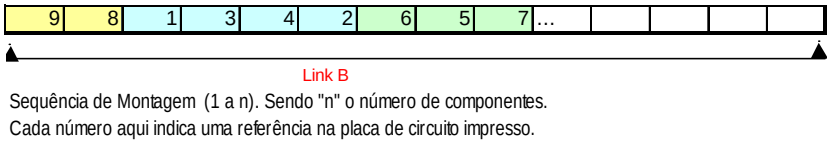


Figura 3.2 – Estrutura do LinkB

3.1.2. Função Objetivo

A fim de avaliar o grau de aptidão dos cromossomos (*LinkA* e *LinkB*) de uma população, foi necessário desenvolver uma função objetivo que considerasse os principais parâmetros físicos da máquina (tempos, velocidade do braço, coordenadas, etc.) e calculasse o tempo de montagem de uma placa a partir das variáveis de entrada: mapa de alimentação (*LinkA*) e a seqüência de montagem (*LinkB*), conforme figura 3.3.

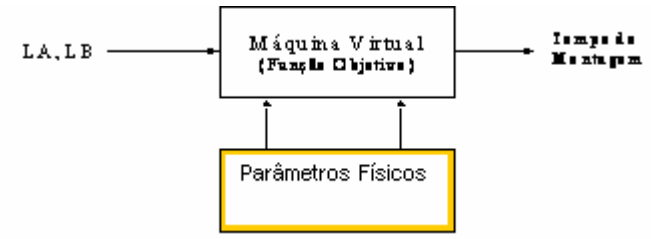


Figura 3.3 – Diagrama da Função Objetivo

Pelo fato de terem sido utilizados parâmetros físicos da máquina na elaboração da função objetivo, a mesma foi denominada neste trabalho de “máquina virtual”.

Em comparação com o modelo formulado na seção 2.3 por Ayob e Kendall [12], Lee et al. [5] propuseram a construção de uma Função Objetivo de forma sintetizada e que inclui o termo referente à troca de *nozzle*. Essa formulação foi, então, adotada na implementação do AG proposto e, referindo-se às variáveis definidas no início desse capítulo, a função objetivo foi definida como:

$$F(\tilde{l},\tilde{l}^*) = \sum_{k=1}^{n}\left[T(l(b(l^*(k))),l^*(k)) + F_1(\tilde{l},\tilde{l}^*,k)\right] \tag{3.1}$$

$$\text{onde } F_1(\tilde{l},\tilde{l}^*,k) = \begin{cases} 0, \text{ se } k = n \\ T(l^*(k),l(b(l^*(k+1)))), \text{ se } b(l^*(k)) = b(l^*(k+1)) \\ T(l^*(k),ANC) + Tanc + T(ANC,l(b(l^*(k+1)))), \text{ c.c.} \end{cases} \tag{3.2}$$

O primeiro termo da equação 3.1 corresponde ao tempo de deslocamento do braço entre o alimentador em que está o carretel com o componente a ser capturado, $l^*(k)$, e a posição na placa onde o mesmo será montado.

O segundo termo corresponde ao tempo de deslocamento do braço entre a posição na placa em que foi montado o componente $l^*(k)$ e o alimentador do próximo componente a ser montado, $l^*(k+1)$, quando o tipo de *nozzle* de $l^*(k)$ **for igual** ao tipo de *nozzle* de $l^*(k+1)$, conforme figura 3.4. De outra forma:

- Será nulo quando $l^*(k)$ for o último componente da seqüência;
- Levará em conta os tempos de ida ao ANC e *Tanc*, quando o tipo de *nozzle* de $l^*(k)$ **for diferente** do tipo de *nozzle* de $l^*(k+1)$, pois o braço precisará passar pelo ANC, efetuar a troca de nozzle e então partir para a captura do próximo componente, $l^*(k+1)$, conforme figura 3.5.

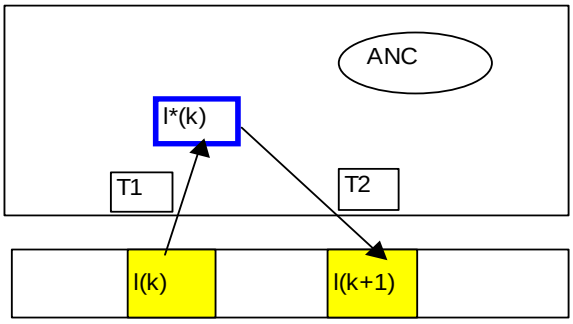


Figura 3.4 – Percurso para componentes que utilizam o mesmo nozzle

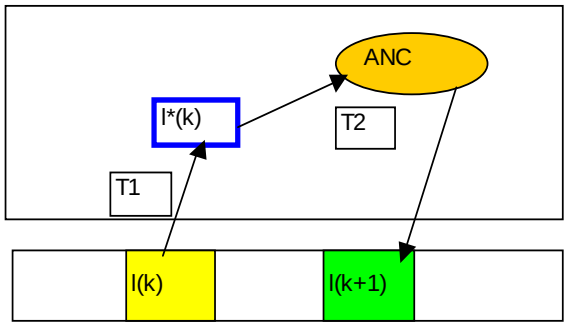


Figura 3.5 – Percurso para componentes que utilizam *nozzles* diferentes

Quanto à **Função de Aptidão**, escolheu-se a técnica da Pontuação (‘*Ranking*’, descrita em maiores detalhes no Apêndice) para transformar os valores encontrados pela Função Objetivo (tempos de montagem) em ‘graus de aptidão’. Na implementação, aproveitou-se a função ‘**ranking**’ do *GA ToolBox* [18] para *Matlab* conforme abaixo:

FitnVx = ranking(X(:, 1));

O único parâmetro da função é:

- ‘X(: , 1)’: vetor com os valores objetivos de todos os indivíduos da população;

O parâmetro MAX, citado no Apêndice, é implícito à função e assume valor 2.

3.1.3. Operadores do AG

No Apêndice desse trabalho encontra-se uma descrição detalhada sobre **operadores genéticos** e como atuam nas estruturas dos cromossomos durante o processo de evolução. Para a estrutura de cromossomos com *links* parcias, utilizando a representação tipo **Rota**, existem diversas técnicas de mutação e cruzamento apropriadas propostas na literatura [16] e [17]. A seguir, serão descritas as que foram utilizadas na implementação do AG.

Mutação

Implementou-se três operadores de mutação: Inversão (figura 3.6), Troca Recíproca (figura 3.7) e Heurístico (figura 3.8):

Inversão:

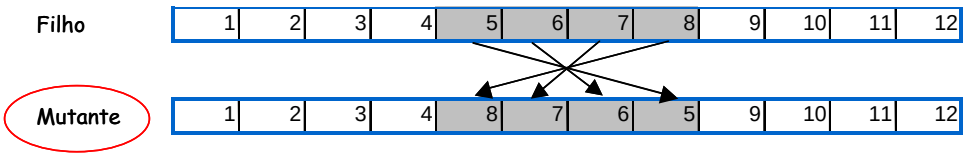


Figura 3.6 – Operador de Mutação (Inversão)

Troca Recíproca:

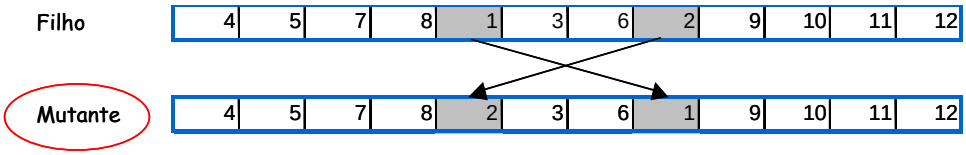


Figura 3.7 – Operador de Mutação (Troca Recíproca)

Heurístico:

O operador de mutação heurístico, NSMλ [19], executa uma busca na vizinhança dos λ genes aleatoriamente escolhidos de um dado cromossomo, conforme procedimento e figura 3.8 abaixo:

- Escolha de λ genes aleatórios (utilizou-se λ = 3)
- Formação dos cromossomos mutantes (*vizinhos*), permutando-se dos λ genes escolhidos (λ! possibilidades).
- Avaliação do grau de aptidão de cada *vizinho* e escolha do melhor para substituir o cromossomo original.

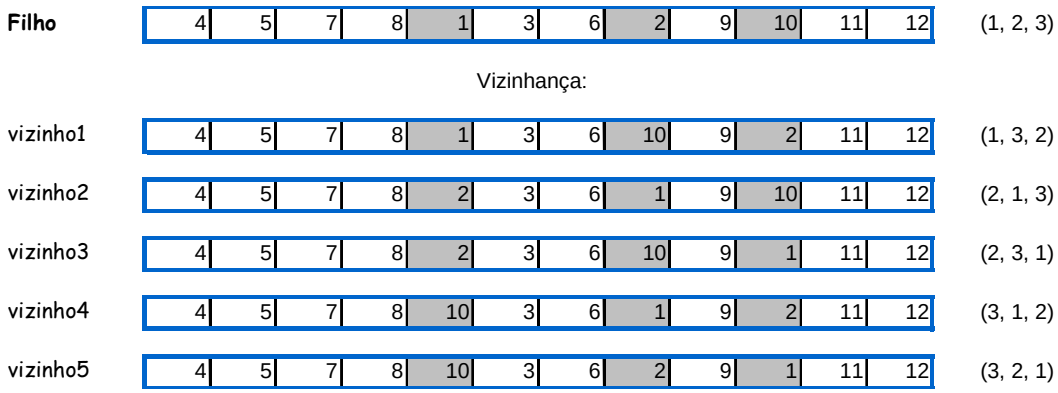


Figura 3.8 – Operador de Mutação (Heurístico)

Cruzamento

O operador de cruzamento escolhido e implementado foi o tipo Ordenado (*OX: Ordered Crossover*).

OX – Cruzamento Ordenado: o procedimento está descrito e ilustrado na figura 3.9 abaixo:

- escolhe-se, aleatoriamente, uma partição de genes do primeiro cromossomo *pai* (a partição é definida por seu tamanho e gene inicial)
- copia-se a partição escolhida para o cromossomo *filho*, na mesma posição
- os demais genes (complementares) do cromossomo filho são herdados do segundo cromossomo *pai*, seguindo a **ordem** em que aparecem da esquerda para a direita, excetuando-se os genes que já constam na partição.
- o procedimento é, então, repetido para a formação do segundo filho, porém herdando-se a partição do segundo *pai* e os genes complementares do primeiro pai.

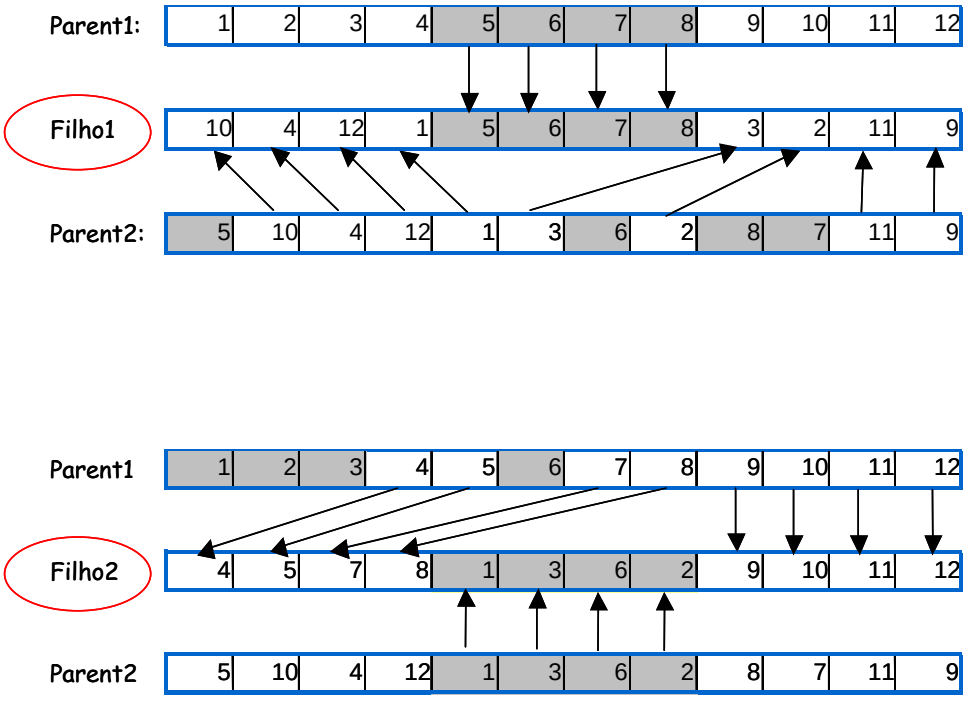


Figura 3.9 – Cruzamento Ordenado

Seleção

Para a seleção adotou-se o método SUS (*Stochastic Universal Sampling*), descrito no Apêndice, com tamanho da população '*n*' denominado variável *pop_size* no programa desenvolvido neste trabalho.

Na implementação, foi utilizada a função '**select**' do *GA ToolBox* [18] para *Matlab* conforme abaixo:

BEST_Parents = select('sus', X, FitnVx, 0.5);

Os parâmetros da função são:

- 'sus': escolha da do método SUS;
- 'X': população de indivíduos (cromossomos);
- 'FitnVx': graus de aptidão de cada cromossomo em 'X';
- GGAP (*generation gap*) = '0.5': parâmetro que determina quantos elementos da população original, X, serão selecionados para próxima geração. Nesse caso, são selecionados *n* x GGAP indivíduos como *BEST_Parents*.

3.2. Caso *Multi-Head* (MH)

A implementação para o caso *Single-Head* descrita na seção anterior tratou os problemas de RAP e PAPSP utilizando a representação de dois cromossomos (LinkA e LinkB) interdependentes. Será apresentado, a seguir, como essa representação foi adaptada para resolver o caso Multi-Head, de forma original, com a inclusão do problema de TAP (*nozzle* assignment) sem restrições à utilização de nozzles repetidos nos cabeçotes.

3.2.1. Formação dos cromossomos – MH

O mapa de alimentação para o caso *Multi-Head*, representado pelo LinkA, utiliza a mesma estrutura de cromossomo do caso *Single-Head* conforme figura 3.10 abaixo, em que cada gene representa o slot do magazine onde será alocado cada tipo de componente:

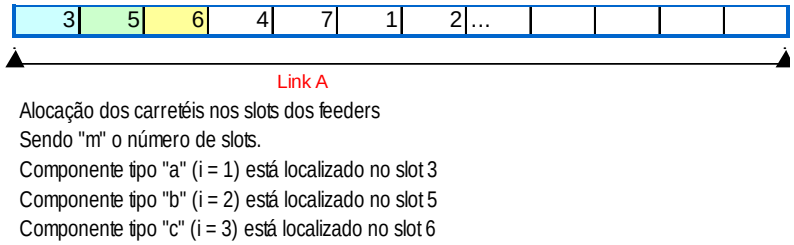


Figura 3.10 – Estrutura do LinkA para o caso *Multi-Head*

O cromossomo que representa a seqüência de montagem dos componentes, *LinkB*, foi adaptado de forma a comportar a associação de *nozzles* por cabeçote e complementado por um cromossomo auxiliar, *LinkB1*, com a finalidade de armazenar a melhor seqüência de *placement* de um ciclo (*tour*). No modelo proposto por Lee et al. [5], existe a restrição de que a ordem em que os componentes são montados na placa de circuito impresso é predeterminada: o cabeçote No. 1 monta seu componente primeiro, o cabeçote No. 2 monta o segundo e assim por diante. O modelo desenvolvido nesse trabalho para máquinas JUKI não possui essa restrição, sendo livre a seqüência de acionamento dos cabeçotes para a montagem dos componentes em um ciclo. A figura 3.11 mostra essa nova estrutura:

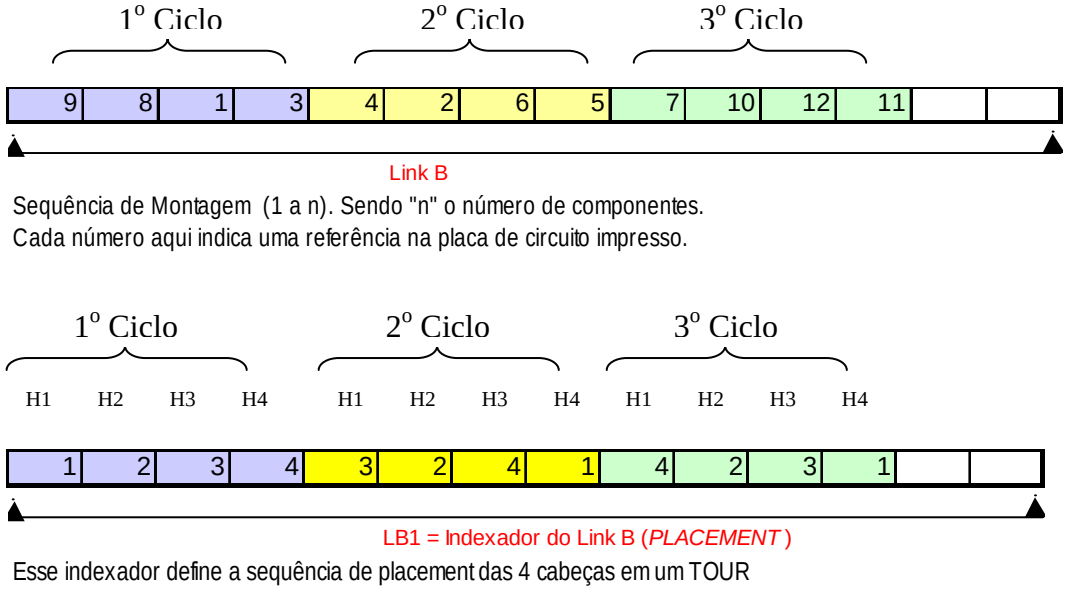


Figura 3.11 – Estrutura dos Links B e B1 para o caso *Multi-Head*

Tratando-se de uma máquina com H = 4 cabeçotes h_i, $i = 1,...,4$, cada ciclo será composto de, no máximo, 4 componentes. A ordem em que aparecem da esquerda para

a direita no *link* B representa a seqüência em que os componentes serão capturados nos alimentadores pelos cabeçotes h_1 a h_4.

Para cada ciclo, surge um sub-problema de TSP com até 4 cidades a serem visitadas e, nesse caso, 24 (= 4!) possibilidades. O *link* B1 exerce a função de indexador do *link* B, armazenando para cada ciclo a seqüência ótima de montagem dos componentes na placa.

A figura 3.12 ilustra para os exemplos de *links* B e B1 acima como fica a disposição dos componentes em cada um dos ciclos de *pick & place*:

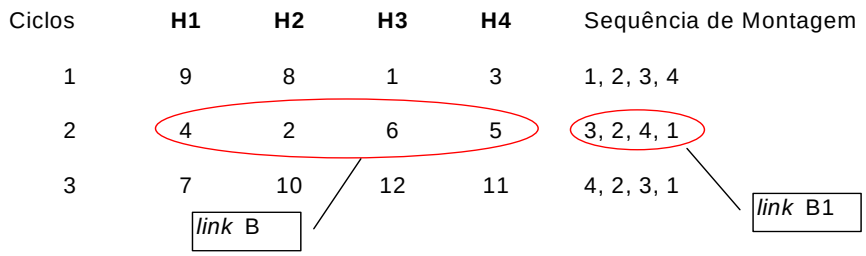


Figura 3.12 – Exemplo do que representam os Links B e B1

Cada um dos n = 12 componentes do exemplo acima possui um tipo de *nozzle* N_i associado que, por sua vez, depende das características mecânicas (peso e dimensão) do componente. Essa associação é pré-definida na biblioteca de componentes conforme abaixo:

link B(i) = { 9, 8, 1, 3, 4, 2, 6, 5, 7, 10, 12, 11 }

N_i = {4, 3, 1, 2, 4, 3, 1, 2, 4, 3, 1, 2 } , sendo:

N_i = tipo de *nozzle* (link B(i))

A partir dessa relação, a solução pode, também, ser representada pelos *nozzles* associados a cada cabeçote, caracterizando o problema de *nozzle assignment* conforme figura 3.13 a seguir:

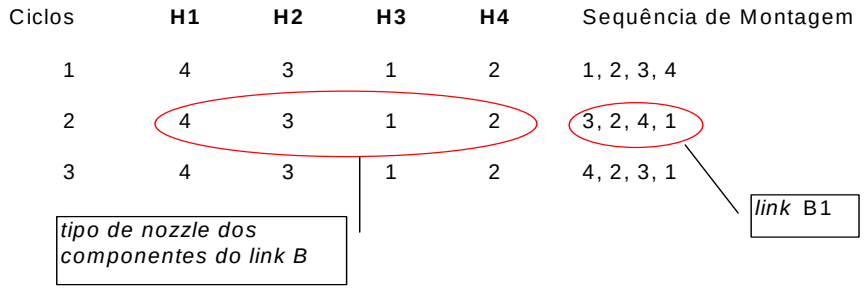


Figura 3.13 – Associação de *nozzles* aos cabeçotes

A representação adotada para os *links* A e B, com indexador B1, no caso *Multi-Head*, permite tratar, de forma simultânea, os diversos problemas de otimização (*nozzle assignment*, *feeder assignment* e *placement*) utilizando os mesmos operadores genéticos de cruzamento, mutação e seleção desenvolvidos para o caso *Single-Head*, haja vista que não houve alteração estrutural nos cromossomos e sim na interpretação dos seus arranjos e na função objetivo (máquina virtual *Multi-Head*) que será detalhada adiante.

3.2.2. Utilização de Nozzles Repetidos - Caso MH-NR

Outra restrição imposta no modelo proposto por LEE et al. [5] é que um determinado tipo de *nozzle* pode estar associado a um, e somente um, cabeçote. Na prática, observamos que essa restrição acarreta em um completo desbalanceamento da carga de trabalho entre os cabeçotes pois, quase sempre, existe a predominância do uso de um tipo de *nozzle* em relação aos demais.

Como exemplo de um caso real, tomamos por base a lista de componentes das placas de circuito impresso de um modelo *de micro-system* de Áudio conforme tabela 3.3.

Observamos, neste exemplo, que existem 261 (75,4%) componentes que utilizam o nozzle tipo “1” enquanto apenas 80 (23,1%), tipo ‘2’, zero (0%), tipo ‘3’, 5 (1,5%), tipo ‘4’.

Após extensa pesquisa na literatura acerca de modelos desenvolvidos para máquinas *multi-head* sem a restrição para o uso de *nozzles* repetidos em mais de um cabeçote, não foi encontrada referência bibliográfica que abordasse o tema.

Na prática, no entanto, verificou-se que o algoritmo utilizado pelo *software* otimizador fornecido pelo fabricante JUKI, denominado HLC (*Host Line Computer*), contempla o caso NR nas soluções propostas.

Para que o AG desenvolvido nesse trabalho resultasse em soluções práticas e competitivas em relação ao otimizador comercial HLC, algumas modificações, além da formação dos cromossomos, foram necessárias e serão descritas nas seções a seguir.

Tabela 3.3 – Exemplo de Lista de Componentes

Tabela de Componentes para *Micro System - Placa Principal*				
CÓDIGO	DESCRIÇÃO	COMP TYPE	QTDE.	NOZZLE TYPE
1-216-833-91	RES, CHIP 10K (1608)	A	48	1
1-216-864-91	CONDUTOR CHIP (1608)	B	44	1
1-216-809-91	RES, CHIP 100 (1608)	C	26	1
1-164-156-91	CAP. CHIP CER. 0.1MF F 1608	E	17	1
1-216-821-91	RES, CHIP 1.0K (1608)	F	15	1
1-216-841-91	RES, CHIP 47K (1608)	G	13	1
1-216-829-91	RES, CHIP 4.7K (1608)	J	10	1
1-216-845-91	RES, CHIP 100K (1608)	K	7	1
1-164-227-91	CAP.CHIP CER. 22000PF B 1608	L	6	1
1-216-827-91	RES, CHIP 3.3K (1608)	N	6	1
1-216-837-91	RES, CHIP 22K (1608)	O	6	1
1-162-970-91	CAP. CHIP CER. 10000PF B 1608	R	4	1
1-164-114-91	CAP.CHIP CER. 390PF B (1608)	S	4	1
1-115-156-91	CAP. CER. 1000000PF F 1608	V	3	1
1-162-923-91	CAP. CER. 47PF CH 1608	X	3	1
1-162-968-91	CAP.CHIP CER. 4700PF B 1608	Z	3	1
1-216-835-91	RES, CHIP 15K (1608)	1B	3	1
1-162-919-91	CAP. CHIP CER 22PF CH 1608	1D	2	1
1-162-946-91	CAP. CHIP CER. 27PF SL 1608	1E	2	1
1-162-960-91	CAP. CHIP CER. 220PF B 1608	1F	2	1
1-162-961-91	CAP. CHIP CER. 330PF B 1608	1G	2	1
1-162-962-91	CAP. CHIP CER. 470PF B 1608	1H	2	1
1-162-964-91	CAP.CHIP CER. 1000PF B 1608	1I	2	1
1-162-966-91	CAP. CER. 2200PF B 1608	1J	2	1
1-216-806-91	RES. CHIP 56 (1608)	1K	2	1
1-216-819-91	RES, CHIP 680 (1608)	1L	2	1
1-216-826-91	RES.CHIP 2,7K 5% 1/16W (1608)	1M	2	1
1-216-832-91	RES, CHIP 8.2K (1608)	1N	2	1
1-216-836-91	RES, CHIP 18K (1608)	1O	2	1
1-216-839-91	RES, CHIP 33K (1608)	1P	2	1
1-216-851-91	RES, CHIP 330K (1608)	1Q	2	1
1-216-853-91	RES, CHIP 470K (1608)	1R	2	1
1-216-857-91	RES, CHIP 1.0M (1608)	1S	2	1
1-218-289-91	RES, CHIP 510 (1608)	1T	2	1
1-218-296-91	RES. CHIP 75K (1608)	1U	2	1
1-113-619-91	CAP. CER. 470000PF F (1608)	1X	1	1
1-162-917-91	CAP. CHIP CER. 15PF CH 1608	1Y	1	1
1-162-953-91	CAP. CHIP CER.100PF SL 1608	1Z	1	1
1-216-813-91	RES, CHIP 220 (1608)	2A	1	1
1-216-815-91	RES, CHIP 330 (1608)	2B	1	1
1-216-843-91	RES, CHIP 68K (1608)	2C	1	1
1-216-861-91	RES, CHIP 2.2M (1608)	2D	1	1
		SUB-TOTAL	261	
1-216-825-91	RES.CHIP 2.2K	D	21	2
8-719-988-61	DIODO 1SS355TE-17	H	13	2
8-729-027-43	TRANSISTOR DTC114EKA-T146	I	13	2
1-216-296-91	RES.CHIP 0 1/8W 5% TC	M	6	2
8-729-802-82	TRANSISTOR 2SC3661	P	6	2
8-729-120-12	TRANSISTOR 2SC1623-T1-L5L6	Q	5	2
1-414-772-21	FERRITE EMI (SMD)	T	4	2
8-729-027-31	TRANSISTOR DTA124EKA-T146	U	4	2
1-216-817-91	RES. CHIP 470	1A	3	2
8-729-120-10	TR.2SA812-T1-M5M6	1C	3	2
8-729-141-73	TRANSISTOR 2SC3624A-T1L15L16	1V	2	2
		SUB-TOTAL	80	
6-702-130-01	IC HA12237F	2E	1	4
6-702-422-11	IC M61519FPD61G	2F	1	4
6-801-837-01	IC M30620MCN-A09FP	2G	1	4
8-759-274-71	IC NJM4565M(TE2)	2H	1	4
8-759-533-04	IC M62703ML-E1	2I	1	4
		SUB-TOTAL	5	
TOTAL:			346	

3.2.3. Função Objetivo para o caso MH

A função objetivo descrita na seção 3.1.2 precisou ser modificada em conseqüência da nova codificação dos cromossomos $\tilde{l}^*$ (*LinkB*) para o caso MH, e adaptada para os termos relacionados ao *pickup*, *placement* e troca de *nozzles*. Como não houve necessidade de alteração no Mapa de Alimentação de componentes, o tratamento aos cromossomos $\tilde{l}$ (*LinkA*) na função objetivo, para esse caso, permaneceu o mesmo.

A equação 3.1 pode então ser reescrita como:

$$F(\tilde{l},\tilde{l}^*) = \sum_{k=1}^{nc} \begin{bmatrix} Tpick(k) + \\ T(l(b(l^*(last\_pick(k))))), l^*(first\_place(k))) \\ Tplace(k) + \\ F_1(\tilde{l},\tilde{l}^*,k) \end{bmatrix}, \qquad (3.3)$$

onde '*nc*' representa o número de ciclos de *pick&place*, e pode ser definido como a divisão inteira do tamanho do *LinkB* pelo número de cabeçotes: *nc* = *ceil*(*length*($\tilde{l}^*$)/*H*). Em Matlab, as funções *ceil* e *length* retornam, respectivamente, o arredondamento de um número real para o inteiro na direção a +∞ e o comprimento do vetor.

Os demais termos da equação são definidos como:

(1)

$$Tpick(k) = \sum_{i=1}^{H-1} \left[T(l(b(l^*((k-1)*H+i))), l(b(l^*((k-1)*H+i+1)))) \right] + Npicks(k)*Tpick,$$

Esse termo representa o tempo de deslocamento do braço para percorrer os *H* alimentadores, do ciclo *k*, somado ao tempo de máquina para uma operação de *pickup* (*Tpick*) multiplicado pelo número de operações de captura (*Npicks*) no ciclo atual, *k*.

Um dos aspectos importantes para a otimização do tempo de montagem em uma máquina MH é maximizar o *pickup* simultâneo dos componentes nos alimentadores. Isso ocorrerá todas as vezes que um alimentador correspondente a um cabeçote for, na

mesma ordem, adjacente ao alimentador do cabeçote vizinho. A figura 3.14 ilustra esse fato com um exemplo em que *Npicks* = 3:

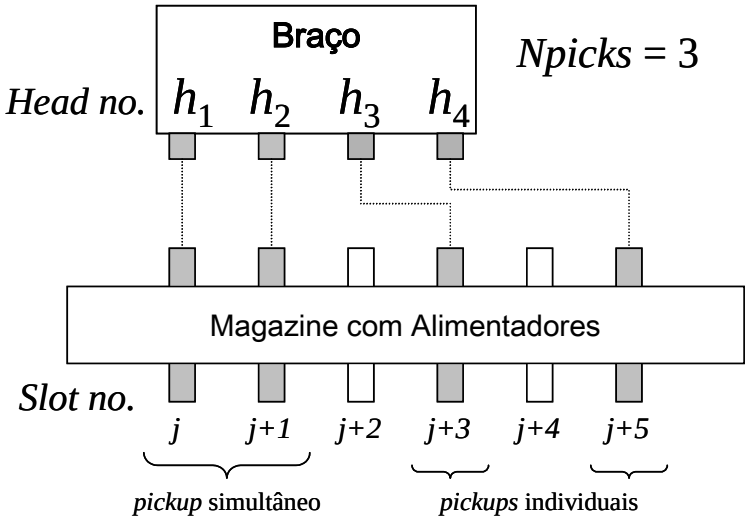


Figura 3.14 – *Pickup* simultâneo entre cabeçotes h_1 e h_2

Sendo assim, o cálculo do termo (1) da equação 3.3 depende da adjacência ou não dos alimentadores em relação aos cabeçotes. Não só o termo *Npicks* será reduzido nesse caso, como também o tempo de deslocamento do braço entre as duas posições vizinhas será nulo: $T(l(b(l^*((k-1)*H+i))), l(b(l^*((k-1)*H+i)))) = 0$.

(2) $T(l(b(l^*(last\_pick(k)))), l^*(first\_place(k)))$,

Esse termo expressa o tempo necessário para o deslocamento do braço entre o último componente capturado nos alimentadores, aqui representado por $l(b(l^*(last\_pick(k))))$, e a primeira posição a ser montada no ciclo *k*, representada por $l^*(first\_place(k))$. Existem algumas sutilezas associadas a esses dois elementos (*last_pick(k), first_place(k)*), importantes para o resultado final, que precisam ser destacadas:

O valor trivial para *last_pick(k),* caso em cada ciclo o braço operasse sempre com os *H* cabeçotes ocupados com componentes, seria $k*H$. Ou seja, os múltiplos inteiros de *H*, no *linkB*, o que corresponde aos últimos elementos capturados em cada

ciclo *k*. No exemplo a seguir os elementos ‘3’, ‘5’ e ‘11’ são, respectivamente, os *last_picks(k)* para *k* = 1, 2 e 3.

Exemplo:

link B(*i*) = { 9, 8, 1, 3, 4, 2, 6, 5, 7, 10, 12, 11 }

Na seção 3.2.5 será vista a importância da inclusão de componentes vazios no *linkB* a fim de ampliar as possibilidades de solução ou espaço de busca. Isso acarretará em ciclos com menos de *H* elementos no *linkB*, surgindo situações em que *last_pick(k)* $\neq k*H$.

O elemento correspondente ao *first_place(k)* também é relativo, pois depende do resultado da otimização do subproblema de TSP com *H* cidades que surgirá no termo (3), abaixo, da equação 3.3. No modelo de Lee et al. [5] , *first_place(k)* = ((*k*-1)**H*+1) pois o primeiro componente a ser montado em cada ciclo sempre será aquele que está no cabeçote h_1.

(3) $Tplace(k) = \min_{\sigma} \sum_{i=1}^{H-1} d_{\sigma(i),\sigma(i+1)}$, onde σ é uma permutação dos *H* elementos de $\tilde{l}^*(k)$. A permutação σ-ótima é denominada na implementação desse trabalho de *LinkB1*. Com isso, é possível concluir que $\tilde{l}^*(first\_place(k)) = \tilde{l}^*((k-1)*H+\sigma(1))$.

(4)

$$F_1(\tilde{l},\tilde{l}^*,k) = \begin{cases} 0, \text{ se } k = Y \\ T(l^*(last\_place(k)), l(b(l^*(first\_pick(k+1))))), \ se: \\ \qquad\qquad b(l^*((k-1)*H+i)) = b(l^*((k)*H+i));\ i = 1,...,H. \\ T(l^*(last\_place(k)), ANC) + Tanc + T(ANC, l(b(l^*(first\_pick(k+1))))), \text{ c.c.} \end{cases}$$

Esse termo estabelece, basicamente, o mesmo que a equação 3.2 para o caso SH, sendo que, para o caso MH, a decisão de ir ou não ao ANC depende de um ou mais dos *H* cabeçotes trocarem de nozzle entre o sub-ciclo atual *k* e o próximo, *k*+1.

Vale ressaltar que independente da quantidade de cabeçotes que deverão trocar de *nozzle* na transição de um ciclo para outro, o termo *Tanc* é constante na equação 3.2, pois o ANC fará a troca de *nozzle* simultaneamente. Daí surge um parâmetro de vital

importância no desempenho do AG: agrupar o máximo de trocas de nozzle possíveis na transição de ciclos.

3.2.4. Problemas com a abordagem simultânea - TAP, RAP e PAPSP

Até esse ponto, o AG proposto trata os três problemas de otimização inter-relacionados de forma simultânea: RAP (*LinkA*), TAP e PAPSP (*LinkB*). Vale ressaltar que o problema de *nozzle assigment*, TAP, por si só, já consiste em um subproblema de otimização multiobjetivo, pois a solução ótima para a alocação de *nozzles* aos cabeçotes deve satisfazer os objetivos abaixo:

- Equalizar a carga de trabalho entre as cabeças, buscando-se o mínimo de ciclos de *pick & place*;
- Minimizar as trocas de *nozzle*;
- Minimizar os deslocamentos ao ANC, priorizando trocas de *nozzle* simultâneas;

A forma descrita anteriormente para a codificação dos cromossomos (seção 3.2.1), assim como a função objetivo implementada (seção 3.2.3) permitem que o AG evolua e chegue a soluções sub-ótimas, porém insatisfatórias em termos de resultados práticos. Os principais problemas detectados a partir de uma análise qualitativa das soluções obtidas com essa versão, comparativamente às soluções propostas pelo HLC, foram:

- convergência prematura com ‘aprisionamento’ do AG em regiões de ótimos locais. Isso devido à forte pressão exercida pelo termo correspondente ao tempo para as trocas de *nozzle*, na função objetivo, quando comparada à pressão exercida pelos demais termos (tempo de deslocamento do braço, temo de *pickup*, etc.)
- a permissão do uso de nozzles repetidos em mais de um cabeçote, visando a equalização da carga de trabalho entre as cabeças, porém sem a inclusão de componentes vazios, resultou em um número elevado e desnecessário de trocas de nozzle;
- o tempo de execução do AG tornou-se alto quando comparado ao tempo de otimização do HLC.

Para solucionar tais problemas e melhorar o desempenho do AG, verificou-se a necessidade de incluir componentes vazios (*blank components*) a fim de tornar o algoritmo mais flexível, ampliando o espaço de busca das soluções e eliminando trocas de *nozzles* desnecessárias. Além disso, resolveu-se desacoplar o problema de TAP dos demais (RAP e PAPSP), tratando-o com uma modelagem genética exclusiva que será descrita adiante.

3.2.5. Inclusão de Componentes Vazios (*'blank' components*)

Antes de descrever a técnica e critérios utilizados para a inclusão de genes 'vazios' na estrutura de cromossomo do *LinkB*, que está associado ao problema de roteamento (PAPSP) e alocação de nozzles aos cabeçotes (TAP), será mostrada a sua importância em termos de ampliação do espaço de busca de soluções. Para isso, tomar-se-á por base o exemplo de cromossomo solução da figura 3.15.

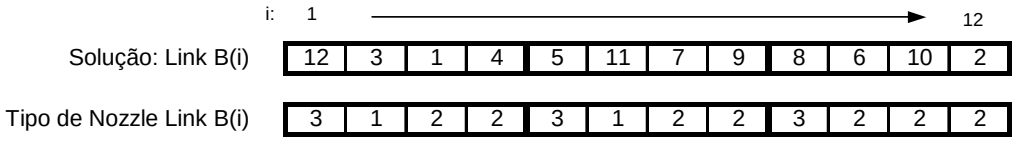


Figura 3.15 – Exemplo de cromossomo solução (LinkB)

Representando-se esse cromossomo solução de forma similar a figura 3.14, em que se destaca a arranjo de *nozzles* por cabeçote, é possível observar uma operação de troca de *nozzle* entre o 2º. e 3º. ciclo no cabeçote h_2, conforme figura 3.16.

Ciclos	H1	H2	H3	H4
1	3	1	2	2
2	3	1	2	2
3	3	2	2	2

Figura 3.16 – Arranjo (I) de *nozzles* por cabeçote do exemplo da fig. 3.15

Aparentemente, essa solução parece boa sob o ponto de vista do problema de TAP: carga de trabalho equalizada entre os cabeçotes, utilização de *nozzles* repetidos

em h_3 e h_4, quantidade mínima de ciclos de *pick&place*., única visita ao ANC e, também, uma única troca de nozzle.

No entanto, na prática, essa solução não é ótima para o TAP do exemplo em questão. As especificações de tempo da máquina JUKI KE-2030 indicam que uma operação de troca de *nozzle* corresponde ao tempo de aproximadamente 5 ciclos de *pick&place*. Essa informação sugere que é vantajoso abrir mão de um ciclo a mais de *pick&place* eliminando a morosa operação de troca de *nozzle*, conforme figura 3.17 abaixo:

Ciclos	H1	H2	H3	H4
1	3	1	2	2
2	3	1	2	2
3	3	X	2	2
4	X	X	2	X

Figura 3.17 – Arranjo (II) de *nozzles* por cabeçote do exemplo da fig. 3.15

O desbalanceamento de carga entre os cabeçotes desse novo arranjo é compensado pela eliminação de operação de troca de *nozzle* no cabeçote h_2. Os elementos identificados por 'X' são vazios, ou seja, no 3º. ciclo o cabeçote h_2 não estará ocupado com nenhum componente, assim como os cabeçotes h_1, h_2 *e* h_4 no 4º. ciclo.

O cromossomo original (*LinkB*) possui apenas 12 genes. Para representar o arranjo com componentes vazios 'X' será necessário ampliá-lo para comportar 16 elementos. Dessa forma, o LinkB com componentes 'vazios' passa a ser representado pela figura 3.18.

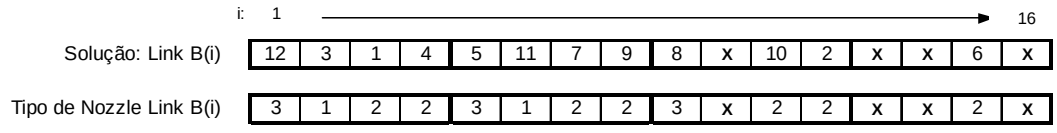


Figura 4.18 – Exemplo de cromossomo acrescido de componentes vazios

As permutações possíveis, ou espaço de busca, dos elementos do *LinkB* aumentaram de 12! para 16!. Isso por um lado, flexibiliza o AG a encontrar soluções ótimas, por outro, acarreta o aumento de seu tempo de execução. Foi estabelecido, portanto, na implementação desse trabalho, um critério para limitar o número de

componentes, ou genes, vazios a serem incluídos no cromossomo correspondente ao *LinkB*:

Critério: a quantidade de ciclos de *pick&place* a serem acrescentados, que contêm componentes vazios, é calculada pela diferença entre a carga de trabalho dos cabeçotes com o maior e menor número de componentes, limitada a 5. Esse valor corresponde exatamente à relação de equivalência de tempo entre 1 troca de *nozzle* e 5 operações de *pick&place* citada anteriormente. Logo, o número de componentes vazios a serem adicionados no *LinkB* corresponde à quantidade necessária para completar o arranjo estendido de nozzles por cabeçotes, de maneira que o mesmo sempre contenha um número total de elementos que seja múltiplo inteiro de *H* (quantidade de cabeçotes).

De antemão, não se conhece qual a diferença de carga de trabalho entre os cabeçotes, pois a otimização e solução do problema de TAP ocorre de forma simultânea aos demais problemas. Com isso, o critério acima descrito só foi possível de ser implementado na prática após o desacoplamento do problema de TAP, descrito na seção 3.2.6.

A presença de componentes vazios no cromossomo *LinkB* não acarretou em necessidade de mudança dos operadores de cruzamento e mutação do AG, porém a Função Objetivo precisou ser alterada para reconhecê-los e tratá-los apropriadamente.

3.2.6. Desacoplamento do problema de *nozzle assignment* – TAP

Face os problemas levantados na seção 3.2.4, resolveu-se desacoplar o problema de TAP dos demais, PAPSP e RAP, mantendo os mesmos objetivos: minimizar a troca de *nozzles*, equalizar a carga de trabalho entre os cabeçotes e reduzir o número de visitas ao ANC maximizando a troca de *nozzles* simultânea. Para isso, foi desenvolvido um AG auxiliar que faz um pré-processamento dos cromossomos *LinkB*, retornando ao AG principal uma nova população inicial com cromossomos pré-otimizados (quanto ao problema de TAP, somente) e já munidos de componentes vazios, para então sofrerem otimização completa: RAP, PAPSP e TAP, inclusive, se necessário. Com isso, manteve-se intacta a implementação do AG principal. A seguir, serão descritas as principais estruturas desenvolvidas para o AG auxiliar que, nesse trabalho, foi denominado de AG-TAP.

Formação dos Cromossomos para o AG-TAP

A população inicial com cromossomos do tipo *LinkB,* $\tilde{l}^*$, associados à seqüência de montagem dos componentes, entra como parâmetro no AG-TAP. A partir de um desses cromossomos, é extraída a informação de quantidade de componentes a serem montados (*n*), assim como os tipos e quantidades de nozzles utilizados nesses componentes. É realizado, então, um novo processo de codificação de cromossomo dando origem a uma representação genética para o arranjo de nozzles por cabeçote, denominado na implementação desse trabalho de *LinkN*.

Esse novo cromossomo possui tamanho *Q* correspondente ao número de posições de um arranjo de nozzles que comporte os *n* componentes da seqüência original mais uma quantidade suficiente de componentes vazios que amplie o espaço de busca segundo o critério estabelecido na seção 3.2.5. Por essa razão, *Q* foi especificado na implementação como: $Q \leq H * (nc + 5)$. Vale ressaltar que para atender o referido critério, tal condição de Q é apenas necessária, porém não suficiente. A condição de suficiência será satisfeita pela restrição imposta na formulação final do problema de otimização do TAP, equação 3.4.

Definido o *LinkN*, é gerada uma população de indivíduos dessa espécie que representa um sub-conjunto das diversas possibilidades de arranjos de *nozzles* para o problema em questão. Ocorre, então, o processo de otimização no AG-TAP, até obter-se o melhor **padrão de arranjo** ('*template*'). Feito isso, decodifica-se o *LinkN* para o formato original de $\tilde{l}^*$, incluindo-se os componentes vazios necessários e $\tilde{l}^*$ passa, então, a ser representado por $\tilde{l}^o$. A figura 3.19 ilustra como seria o formato do *LinkN* que resultaria no cromossomo pré-otimizado do exemplo da seção 3.2.5, figura 3.18.

O AG-TAP retorna para o AG principal uma população de indivíduos do tipo $\tilde{l}^o$ que são **permutações dos genes de mesmo *nozzle*** dentro do padrão de arranjo ótimo encontrado.

Uma das vantagens desse processo desacoplado, ilustrado na figura 3.20, em relação à abordagem simultânea é a drástica redução no tempo de convergência do AG principal, em decorrência do mesmo iniciar a busca a partir de uma população de cromossomos pré-otimizados.

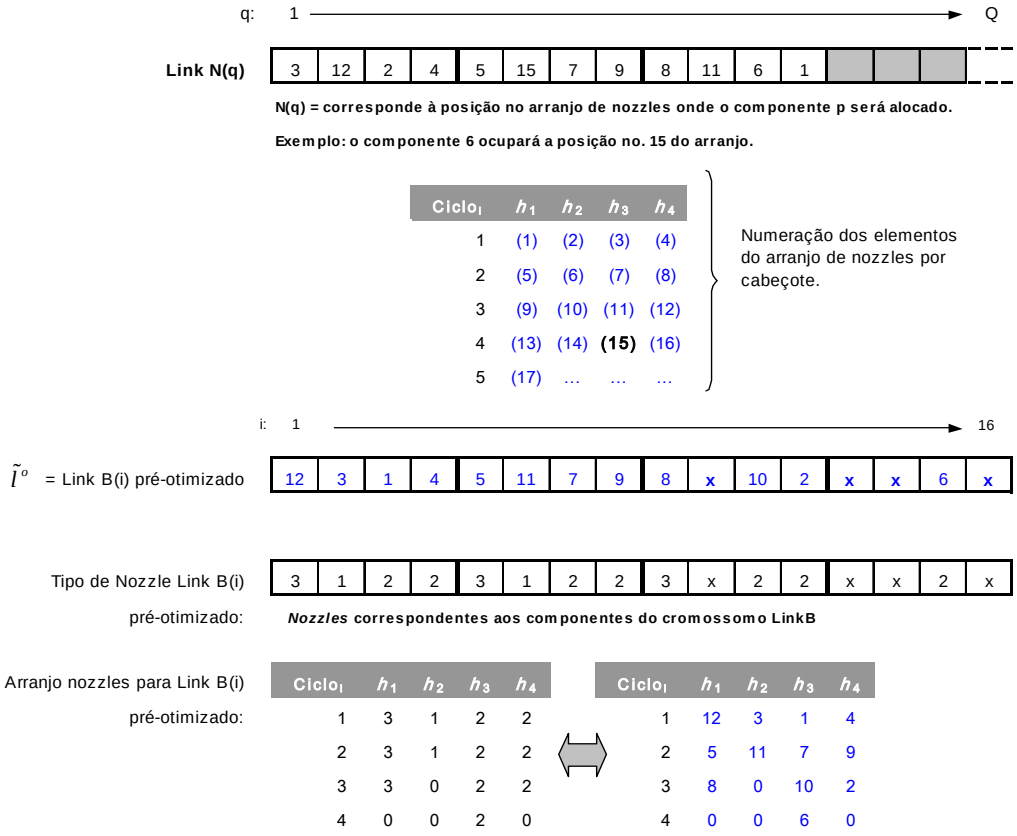


Figura 3.19 – Transformação do *LinkN* em $\tilde{l}^{o}$

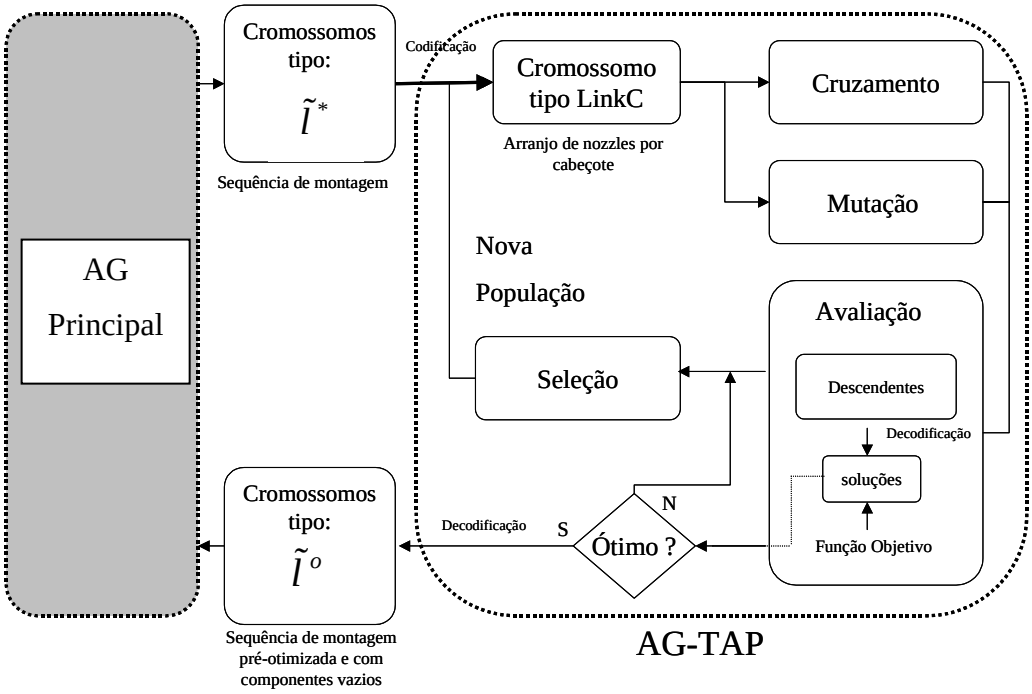


Figura 3.20 – Estrutura do AG-TAP

Operadores do AG-TAP

Os operadores de cruzamento e mutação utilizados nos cromossomos do AG-TAP foram os mesmos do AG principal, respectivamente, cruzamento ordenado, inversão e troca recíproca descritos no Apêndice. O operador de mutação heurística, porém, não foi aproveitado, pois a função de aptidão para o problema de TAP é diferente daquela utilizada no AG principal ('máquina virtual') e, também, pelo fato dos dois operadores de mutação terem-se mostrado suficientes e eficazes para a convergência do AG-TAP. O método de seleção adotado também foi o mesmo (SUS) citado na seção 3.1.3.

Função Objetivo do AG-TAP

Foi criada uma única função objetivo que recebe como parâmetro o cromossomo $\tilde{l}^*$ modificado, $\tilde{l}^o$, a partir de uma soma ponderada de cada um dos objetivos do problema do TAP (*Weighted Sum Method*) [20], de uma forma mais simplificada do que a equação 3.3 do AG principal. Tais objetivos são:

(1) Minimizar o número de ciclos de *pick&place*, de forma a equalizar a carga de trabalho entre os cabeçotes, sujeito a diferença máxima de 5 ciclos de montagem entre os cabeçotes com maior e menor carga de trabalho, pelos motivos descritos na seção anterior;

(2) Minimizar as trocas de *nozzle*;

(3) Minimizar os deslocamentos ao ANC, priorizando trocas de *nozzle* simultâneas.

Conforme exemplo apresentado na seção 3.2.5, os objetivos (1) e (2) podem se tornar conflitantes na busca do melhor arranjo de *nozzles* nos cabeçotes, dependendo dos dados do problema. Para isso, foram adotados pesos diferenciados nos termos da equação 3.4 definidos a seguir:

$\tilde{l}^o$: cromossomo $\tilde{l}^*$ modificado para AG-TAP

α : peso atribuído ao termo *TTN*($\tilde{l}^o$), associado ao objetivo (1)

β : peso atribuído ao termo $TNS(\tilde{l}^o)$, associado ao objetivo (1)

γ : peso atribuído ao termo $DCT(\tilde{l}^o)$, associado ao objetivo (2)

δ : peso atribuído ao termo $ENC(\tilde{l}^o)$, associado ao objetivo (1)

$TTN(\tilde{l}^o)$: quantidade total de trocas de *nozzle* em uma solução $\tilde{l}^o$

$TNS(\tilde{l}^o)$: quantidade de trocas de *nozzle* simultâneas em uma solução $\tilde{l}^o$

$DCT(\tilde{l}^o)$: diferença da carga de trabalho entre os cabeçotes (Max – Min)

$ENC(\tilde{l}^o)$: termo auxiliar que indica o espalhamento de *nozzles* entre os cabeçotes

A Função Objetivo do AG-TAP ficou assim definida:

$$F(\tilde{l}^o) = \alpha * TTN(\tilde{l}^o) - \beta * TNS(\tilde{l}^o) + \gamma * DCT(\tilde{l}^o) + \delta * ENC(\tilde{l}^o),$$

com: $\alpha = 2$, $\beta = 0.50$, $\delta = 0.45$ e $\gamma = 0.05$.

E o problema de otimização do TAP pode, então ser escrito como:

$$\begin{aligned} &min\ F(\tilde{l}^o) = \alpha * TTN(\tilde{l}^o) - \beta * TNS(\tilde{l}^o) + \gamma * DCT(\tilde{l}^o) + \delta * ENC(\tilde{l}^o), \\ &\text{s.t. } DCT(\tilde{l}^o) \leq 5 \end{aligned} \tag{3.4}$$

Até esse ponto do trabalho, a formulação do problema de otimização havia sido feita sem uso de restrições (*unconstrained optimization*). No entanto, o objetivo (1) impõe explicitamente uma restrição na equação 3.4, que reduz o espaço de busca, sendo necessário modificar a representação do problema para: min $f(x)$, s.t. $x \in \Omega$ [16]. No AG-TAP foi implementada a restrição acima, eliminando do processo de seleção todos os cromossomos que não atendem à mesma. Isso foi feito atribuindo-se diretamente um valor indesejável para o grau de aptidão desses indivíduos.

Os valores dos pesos foram atribuídos, inicialmente, conforme a relevância de cada termo da equação 3.4 e ajustados, gradualmente, durante os experimentos até que se obtivessem resultados satisfatórios. Vale observar que o peso β possui sinal trocado pois visa privilegiar o termo $TNS(\tilde{l}^o)$ conforme estabelecido no objetivo (1).

O termo auxiliar $ENC(\tilde{l}^o)$ foi incluído na função com o objetivo de permitir o escape de ótimos locais que não são afetados pelos termos $TTN(\tilde{l}^o)$ e $TNS(\tilde{l}^o)$.

A Função de Aptidão utilizada foi a ‘ranking’ descrita na seção 3.1.2.

Capítulo 4

Resultados

Diversos experimentos práticos, inclusive em campo utilizando uma máquina JUKI KE-2030, foram realizados para avaliação, melhoria e validação do AG implementado nesse trabalho.

No entanto, antes de apresentar os resultados obtidos e análise comparativa das soluções propostas pelo AG e HLC, serão feitos alguns comentários acerca do tempo de execução do algoritmo e análise da contribuição individual das principais rotinas desenvolvidas para o AG.

4.1. Avaliação do tempo de execução do AG

O AG foi executado exaustivamente no PC descrito na seção 1.4, para um exemplo de placa pequena com apenas 21 componentes [5] a fim de testar seu funcionamento para o caso mais simples, SH, e avaliar o tempo total de processamento.

Inicialmente, para uma população de 150 indivíduos e apenas 50 gerações, o tempo de execução do AG foi de 281s. Para melhorar esse desempenho, o código fonte das rotinas foi modificado, aplicando o recurso de **vetorização** do Matlab que permite eliminar as operações de *loop* (*for* e *while*) consideradas ineficientes sob o ponto de vista de tempo computacional.

O tempo foi drasticamente reduzido para 96s, o que representa aproximadamente 1/3 do tempo anterior, como ilustra a figura 4.1.

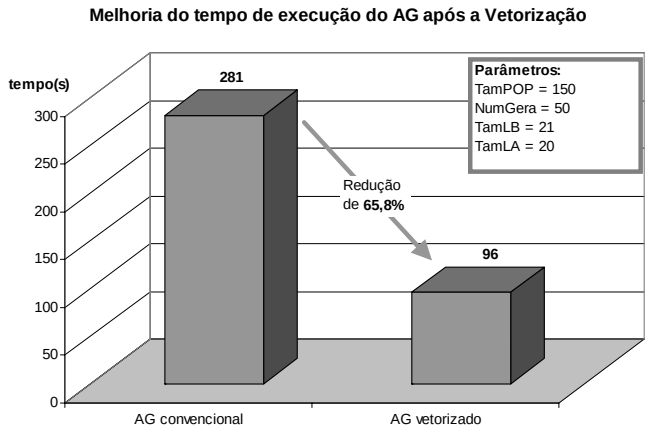


Figura 4.1 – Resultado após a vetorização do AG

Já com o código fonte otimizado por meio da vetorização e utilizando a funções *clock* e *etime* (*elapsed time*) do Matlab, foi possível fazer uma análise completa dos tempos do AG, a fim de identificar quais as rotinas que carregam mais o processador durante a execução do programa. A figura 4.1 traduz esses resultados sob a forma de um gráfico de Pareto:

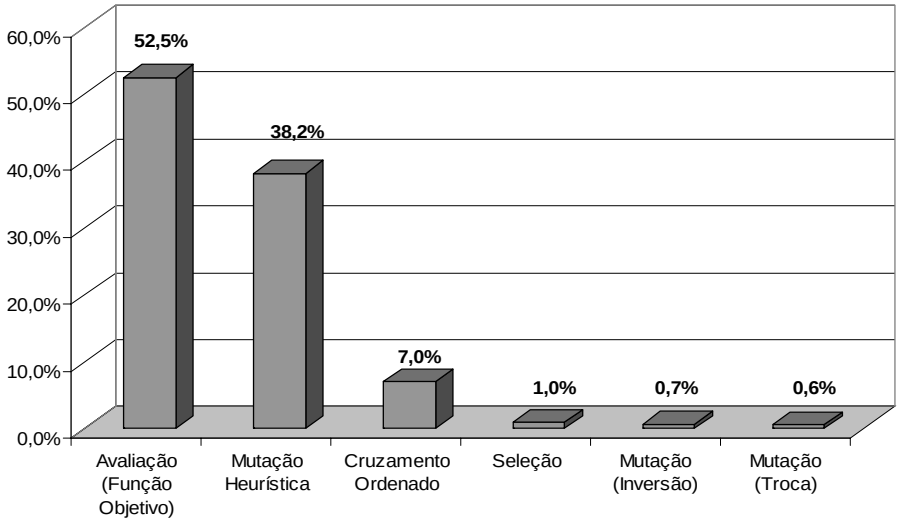


Figura 4.2 – Análise de tempos do AG

Era esperado que a função objetivo (‘máquina virtual’) fosse responsável pela maior ocupação do processador (52,5%), tendo em vista a complexidade da sua formulação e o número de vezes que é chamada em cada geração.

No entanto, surpreendentemente, observa-se que a mutação heurística, NSMλ, ficou em segundo lugar, responsável por 38,2% do tempo total do AG , quando executada com λ = 3 e um percentual de 1% (*pm* = 0.01). O tamanho da população, *n*, utilizado nos ensaios foi de 100 indivíduos. Com esses parâmetros, um único indivíduo (*n*pm*) necessariamente passa pelo processo de mutação heurística a cada geração, o que implica na chamada a função objetivo λ! vezes a fim de encontrar a melhor permutação de genes. Destaca-se, ainda, que como cada indivíduo é composto por 2 cromossomos parcias, *Links* A e B, isso resulta num total de λ! * λ! possibilidades. Apesar desse custo, a função de mutação heurística foi mantida no AG pelo seu importante papel nas características de convergência do AG e na diversidade da população.

Após as análises de tempo acima, o AG foi submetido a problemas de maior porte, tais como o exemplo da figura 4.4 que será detalhado na seção 4.2. Mesmo assim, mostrou-se apto a resolver tal problema *Multi-Head* com 750 gerações, tamanho da população igual a 100, 14 carretéis de componentes e um número de posições de montagem igual a 44, em apenas 26 minutos.

A partir da seção 4.2, serão apresentados os dados completos de um problema exemplo que foi utilizado para realizar testes em campo para fins de comparação do AG proposto com o programa comercial fornecido pelo fabricante das máquinas JUKI: o HLC.

4.2. Dados do problema exemplo

Os dados necessários para gerar um programa para uma máquina de montagem automática são: IFF e BOM. O IFF (*Information File Format*) é um arquivo que traz informações sobre a posição (coordenadas x, y) dos componentes na placa de circuito impresso em relação a um referencial, origem, denominado de fiducial. O B.O.M. (*Bill of Materials*), por sua vez, indica quais os componentes (*part numbers*) que devem ser montados em cada posição da placa.

Para o problema exemplo, a figura 4.3 ilustra a disposição física dos componentes na placa de circuito impresso a partir das informações extraídas do arquivo de coordenadas IFF.

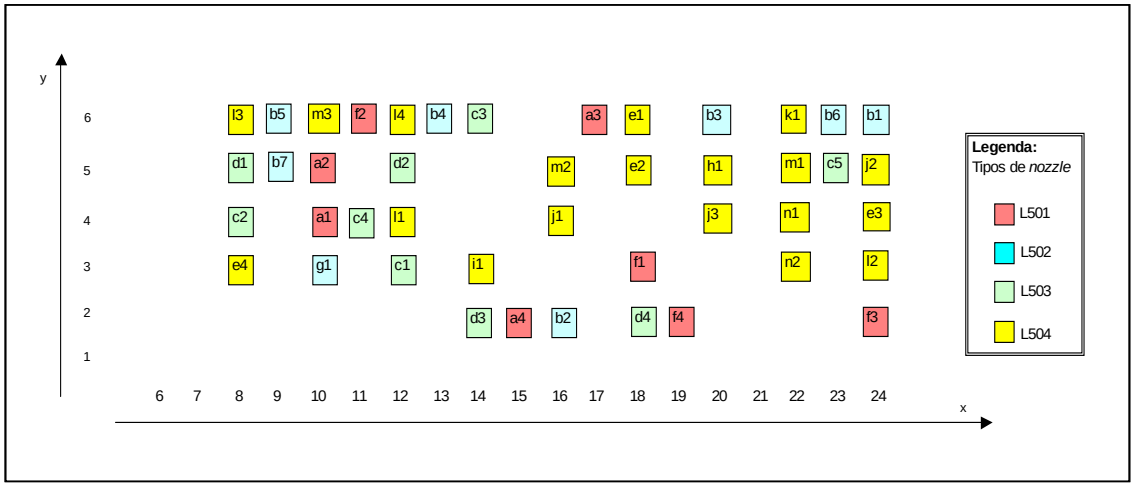


Figura 4.3 – Disposição física dos componentes na PCI

Na tabela 4.1 encontra-se a lista de componentes a serem montados na placa de circuito impresso e seus respectivos *nozzles*, obtida do arquivo B.O.M (*Bill of Materials*) e da biblioteca de componentes da máquina JUKI KE-2030.

Tabela 4.1 – B.O.M e tipos de *nozzle* para o problema exemplo.

Z	CÓDIGO	CompType	NozzleType	DESCRIÇÃO	QTY	REFERÊNCIA
0	1-688-080-12					
	1-162-923-91	1	L(501)		4	a1, a2, a3, a4
	1-218-289-91	2	L(502)		7	b1, b2, b3, b4, b5, b6, b7
	1-216-186-91	3	L(503)		5	c1, c2, c3, c4, c5
	1-216-809-91	4	L(503)		4	d1, d2, d3, d4
	8-729-920-75	5	L(504)		4	e1, e2, e3,e4
	1-216-067-91	6	L(501)		4	f1, f2, f3, f4
	1-216-833-91	7	L(502)		1	g1
	1-469-152-21	8	L(504)		1	h1
	8-719-069-60	9	L(504)		1	i1
	1-242-772-91	10	L(504)		3	j1, j2, j3
	8-729-424-02	11	L(504)		1	k1
	1-125-891-91	12	L(504)		4	l1, l2, l3, l4
	8-719-058-24	13	L(504)		3	m1, m2, m3
	8-729-920-86	14	L(504)		2	n1, n2
				Total:	44	

4.3. Resultados do HLC

Os dados da seção 4.2 foram cadastrados em um arquivo de teste no software de otimização comercial HLC (`testecarlos.H51`). Após configuração dos parâmetros do simulador, a otimização foi realizada e os resultados obtidos foram listados em dois

arquivos textos representando respectivamente a solução para o *placement & nozzle assignment* (PAPSP e TAP) e *feeder assignment* (RAP).

4.3.1. HLC: Placement & Nozzle Assignment

Na figura 4.4, visualiza-se o arquivo texto (parte I) correspondente ao programa de máquina otimizado pelo HLC para os dados do problema exemplo, com a seqüência de montagem dos componentes (coluna 'input') e associação de *nozzles* aos cabeçotes (coluna 'Ctrg').

```
--- Production program ---                      10/28/2004 15:46   Page    1

< Placement data/opt. seq. >                    Line name  LINHAS

Program name = testecarlos.H51            10/28/2004 15:45
PWB ID       = KV-29FA210 A
User ID      =
Station      = #1 JUKI1          (2030)
Mode Specification = Parallel Mode
No.(input) CompID      X       Y    Angle  Compo name          Ctrg.   Head Ckt. p  Try  L
  1     2  A4         15.00    2.00   0.00 1-162-923-91        L(501)  H1      1    No   4
  2    30  F3         24.00    2.00   0.00 1-216-067-91        L(501)  H2      1    No   4
  3    43  G1         10.00    3.00   0.00 1-216-833-91        L(502)  H3      1    No   4
  4    39  B1         24.00    6.00   0.00 1-218-289-91        L(502)  H4      1    No   4
  5     9  A1         10.00    4.00   0.00 1-162-923-91        L(501)  H1      1    No   4
  6    20  F4         19.00    2.00   0.00 1-216-067-91        L(501)  H2      1    No   4
  7    37  B6         23.00    6.00   0.00 1-218-289-91        L(502)  H3      1    No   4
  8     6  B3         20.00    6.00   0.00 1-218-289-91        L(502)  H4      1 /  No   4
  9    10  A2         10.00    5.00   0.00 1-162-923-91        L(501)  H1      1    No   4
 10    16  F2         11.00    6.00   0.00 1-216-067-91        L(501)  H2      1    No   4
 11     5  B2         16.00    2.00   0.00 1-218-289-91        L(502)  H3      1    No   4
 12    35  B4         13.00    6.00   0.00 1-218-289-91        L(502)  H4      1 /  No   4
 13     1  A3         17.00    6.00   0.00 1-162-923-91        L(501)  H1      1    No   4
 14    25  F1         18.00    3.00   0.00 1-216-067-91        L(501)  H2      1    No   4
 15    33  B5          9.00    6.00   0.00 1-218-289-91        L(502)  H3      1    No   4
 16    11  B7          9.00    5.00   0.00 1-218-289-91        L(502)  H4      1 /  No   4
 17    23  D4         18.00    2.00   0.00 1-216-809-91        L(503)  H4      1    No   4
 18    32  C2          8.00    4.00   0.00 1-216-186-91        L(503)  H3      1    No   4
 19    12  M3         10.00    6.00   0.00 8-719-058-24        L(504)  H2      1    No   4
 20     3  J1         16.00    4.00   0.00 1-242-772-91        L(504)  H1      1 /  No   4
 21    18  D3         13.00    2.00   0.00 1-216-809-91        L(503)  H4      1    No   4
 22    36  C4         11.00    4.00   0.00 1-216-186-91        L(503)  H3      1    No   4
 23     4  M2         16.00    5.00   0.00 8-719-058-24        L(504)  H2      1    No   4
 24     8  J3         20.00    4.00   0.00 1-242-772-91        L(504)  H1      1 /  No   4
 25    28  D2         12.00    5.00   0.00 1-216-809-91        L(503)  H4      1    No   4
 26    34  C1         12.00    3.00   0.00 1-216-186-91        L(503)  H3      1    No   4
 27     7  M1         22.00    5.00   0.00 8-719-058-24        L(504)  H2      1    No   4
 28    13  J2         24.00    5.00   0.00 1-242-772-91        L(504)  H1      1 /  No   4
 29    15  E4          8.00    3.00   0.00 8-729-920-75        L(504)  H1      1    No   4
 30    29  E3         24.00    4.00   0.00 8-729-920-75        L(504)  H2      1    No   4
 31    40  C5         23.00    5.00   0.00 1-216-186-91        L(503)  H3      1    No   4
 32    38  C3         13.00    6.00   0.00 1-216-186-91        L(503)  H4      1 /  No   4
 33    19  E2         18.00    5.00   0.00 8-729-920-75        L(504)  H1      1    No   4
 34    24  E1         18.00    6.00   0.00 8-729-920-75        L(504)  H2      1    No   4
 35    14  D1          8.00    5.00   0.00 1-216-809-91        L(503)  H4      1 /  No   4
 36    31  L3          8.00    6.00   0.00 1-125-891-91        L(504)  H2      1    No   4
 37    26  L4         12.00    6.00   0.00 1-125-891-91        L(504)  H1      1 /  No   4
 38    44  L1         12.00    4.00   0.00 1-125-891-91        L(504)  H2      1    No   4
 39    21  L2         24.00    3.00   0.00 1-125-891-91        L(504)  H1      1 /  No   4
 40    22  N2         22.00    3.00   0.00 8-729-920-86        L(504)  H1      1    No   4
 41    27  N1         22.00    4.00   0.00 8-729-920-86        L(504)  H2      1 /  No   4
 42    17  I1         13.00    3.00   0.00 8-719-069-60        L(504)  H2      1    No   4
 43    41  K1         22.00    6.00   0.00 8-729-424-02        L(504)  H1      1 /  No   4
 44    42  H1         20.00    5.00   0.00 1-469-152-21        L(504)  H1      1 /  No   4
```

Figura 4.4 – Programa de montagem otimizado pelo HLC (Parte I)

Sintetizou-se os dados do arquivo texto da figura 4.3 na representação da tabela 4.2 a fim de ilustrar os resultados em termos de ciclos de montagem e alocação de *nozzles* aos cabeçotes, o que permite uma fácil visualização qualitativa da solução:

Tabela 4.2 – Representação dos dados do programa otimizado pelo HLC (parte I)

Ciclo	H1	H2	H3	H4	Posições	*Feeders*	*Head placement sequence*
1	1	1	2	2	a4 f3 g1 b1	36 38 40 42	3 2 1 4
2	1	1	2	2	a1 f4 b6 b3	36 38 42 42	1 4 3 2
3	1	1	2	2	a2 f2 b2 b4	36 38 42 42	1 2 4 3
4	1	1	2	2	a3 f1 b5 b7	36 38 42 42	2 1 3 4
5	4	4	3	3	j1 m3 c2 d4	20 22 24 26	2 3 4 1
6	4	4	3	3	j3 m2 c4 d3	20 22 24 26	4 3 2 1
7	4	4	3	3	j2 m1 c1 d2	20 22 24 26	3 4 1 2
8	4	4	3	3	e4 e3 c5 c3	16 18 24 24	1 4 3 2
9	4	4		3	e2 e1 d1	16 18 26	3 2 1
10	4	4			l4 l3	08 10	1 2
11	4	4			l2 l1	08 10	2 1
12	4	4			n2 n1	48 50	1 2
13	4	4			k1 i1	04 06	1 2
14	4				h1	02	1

- A primeira coluna corresponde aos ciclos de pick&place.
- As colunas H1 a H4 indicam os cabeçotes e os seus respectivos tipos de *nozzles* para cada ciclo de pick & place. Por exemplo, no primeiro ciclo os cabeçotes H1 e H2 estarão munidos de nozzles do tipo ‘1’ (na prática, ‘L(501)’) e os cabeçotes H3 e H4, do tipo ‘2’ (‘L(502)’);
- A coluna ‘Posições’ indica quais os componentes que serão montados em cada ciclo e suas respectivas posições;
- A coluna *Head Placement Sequence* indica a seqüência de acionamento dos cabeçotes (e consequentemente de visita às cidades) que representa o caminho mínimo entre as posições a serem montadas em um determinado ciclo. Observe, na figura 4.5, para o ciclo No. 1, como ficam os percursos sobre a PCI para os casos de acionamento:
  (1) pré-determinado [5]: (H1–H2–H3–H4 $\Leftrightarrow$ a4$\rightarrow$f3$\rightarrow$g1$\rightarrow$b1);
  (2) livre e otimizado: (H3–H2–H1–H4 $\Leftrightarrow$ g1$\rightarrow$f3$\rightarrow$a4$\rightarrow$b1);

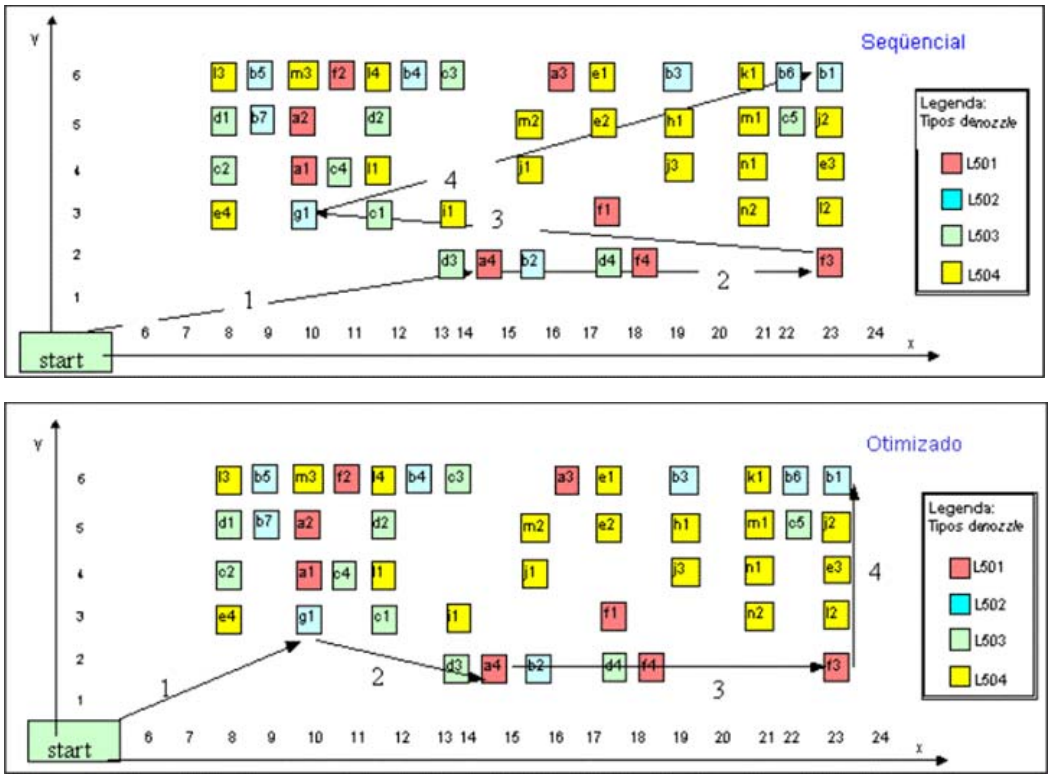


Figura 4.5 – Percursos com acionamento sequencial e otimizado

Quatro aspectos qualitativos da solução encontrada pelo HLC, para os sub-problemas de *placement* e *nozzle assignment*, podem ser depreendidos diretamente da tabela 4.2, como:

- Total de Ciclos = 14
- Número de trocas de *nozzle* = 4
- Número de visitas ao ANC = 1 (pois a troca dos 4 *nozzles* ocorre de forma simultânea entre os ciclos 4 e 5)
- Desbalanceamento da carga de trabalho entre as cabeças: 6 (H1 = 14, H2 = 13, H3 = 8, H4 = 9)

Foi utilizado como parâmetro de avaliação do desbalanceamento entre as cabeças a diferença entre o máximo e o mínimo de componentes alocados aos 4 cabeçotes, nesse caso: | Max – Min | = | 14 – 8 | = 6.

4.3.2. HLC: Feeder Assignment

Na figura 4.6, visualiza-se o arquivo texto (parte II) correspondente ao programa de máquina otimizado pelo HLC para os dados do problema exemplo, com a alocação dos carretéis de componentes nos slots do magazine (RAP), ou mapa de alimentação.

A coluna '*component name*' e a '*sply*' identificam, respectivamente, o tipo de componente (pelo *part number*) e o slot em que o alimentador deste deve ser alocado.

```
--- Production program ---                    10/28/2004 15:46   Page    2

< Pick data/input seq. >                      Line name  LINHAS

Program name = testecarlos.H51           10/28/2004 15:45
PWB ID       = KV-29FA210 A
User ID      =
Station      = #1 JUKI1          (2030)

No. Component name       Pack  Sply    Type   Angle   X1      Y1     Z  Used
  1 1-218-289-91         Tape  F-42    ***        0  353.40    5.00  -0.20 Yes
  2 1-216-186-91         Tape  F-24    ***        0  200.40    5.00  -0.44 Yes
  3 8-729-920-75         Tape  F-16    ***        0  132.40    5.00  -1.00 Yes
  4 8-729-920-75         Tape  F-18    ***        0  149.40    5.00  -1.00 Yes
  5 1-216-809-91         Tape  F-26    ***        0  217.40    5.00  -0.55 Yes
  6 1-216-067-91         Tape  F-38    ***        0  319.40    5.00  -0.42 Yes
  7 1-162-923-91         Tape  F-36    ***        0  302.40    5.00  -0.18 Yes
  8 1-125-891-91         Tape  F- 8    ***        0   64.40    5.00  -0.17 Yes
  9 1-125-891-91         Tape  F-10    ***        0   81.40    5.00  -0.17 Yes
 10 8-719-058-24         Tape  F-22    ***        0  183.40    5.00  -1.00 Yes
 11 1-242-772-91         Tape  F-20    ***        0  166.40    5.00  -1.00 Yes
 12 8-729-920-86         Tape  F-48    ***        0  404.40    5.00  -1.00 Yes
 13 8-729-920-86         Tape  F-50    ***        0  421.40    5.00  -1.00 Yes
 14 8-729-424-02         Tape  F- 4    ***        0   30.40    5.00  -1.00 Yes
 15 8-719-069-60         Tape  F- 6    ***        0   47.40    5.00  -1.00 Yes
 16 1-469-152-21         Tape  F- 2    ***        0   13.40    5.00  -0.19 Yes
 17 1-216-833-91         Tape  F-40    ***        0  336.40    5.00  -0.52 Yes
```

Figura 4.6 – Programa de montagem otimizado pelo HLC (Parte II)

Tomando-se por base a solução acima para o arranjo dos componentes nos alimentadores do magazine (*feeder assignment*), é possível complementar a representação visual da solução, adicionando-se a coluna *feeders*, conforme a tabela 4.3 abaixo:

Tabela 4.3 – Representação dos dados do programa otimizado pelo HLC (partes I e II)

Ciclo	H1	H2	H3	H4	Posições	*Feeders*	*Head placement sequence*
1	1	1	2	2	**a4 f3 g1 b1**	**36 38 40 42**	4 3 1 2
2	1	1	2	2	**a1 f4 b6 b3**	**36 38 42 42**	1 4 3 2
3	1	1	2	2	**a2 f2 b2 b4**	**36 38 42 42**	1 2 4 3
4	1	1	2	2	**a3 f1 b5 b7**	**36 38 42 42**	2 1 3 4
5	4	4	3	3	**j1 m3 c2 d4**	**20 22 24 26**	2 3 4 1
6	4	4	3	3	**j3 m2 c4 d3**	**20 22 24 26**	4 3 2 1
7	4	4	3	3	**j2 m1 c1 d2**	**20 22 24 26**	3 4 1 2
8	4	4	3	3	**e4 e3 c5 c3**	**16 18 24 24**	1 4 3 2
9	4	4		3	**e2 e1 d1**	**16 18 26**	3 2 1
10	4	4			**l4 l3**	**08 10**	1 2
11	4	4			**l2 l1**	**08 10**	2 1
12	4	4			**n2 n1**	**48 50**	1 2
13	4	4			**k1 i1**	**04 06**	1 2
14	4				**h1**	**02**	1

É possível identificar, considerando que *slots* adjacentes são espaçados de 2 *feeders*, os seguintes aspectos qualitativos da solução:

- Número de alimentadores utilizados: 17 (observa-se que a solução proposta pelo HLC não restringe a duplicidade de alimentadores com o mesmo tipo de componente no magazine. Nesse exemplo, três tipos de componentes possuem alimentadores em duplicidade, são eles: componentes “e”, “l” e “n”, respectivamente: 8-729-920-75, 1-125-891-91 e 8-729-920-86. Conforme hipóteses assumidas no item 1.1 desse trabalho, as soluções propostas pelo AG implementado não permitem essa duplicidade, o que representa, de antemão, uma desvantagem em relação ao HLC;
- Número de *pickups* simultâneos com 4 cabeçotes: 4 (ciclos 1, 5, 6, e 7);
- Número de *pickups* simultâneos com 3 cabeçotes: 4 (ciclos 2, 3, 4 e 8);
- Número de *pickups* simultâneos com 2 cabeçotes: 5 (ciclos 9, 10, 11, 12 e 13);
- Número de *pickups* não simultâneos: 1 (ciclo 14);

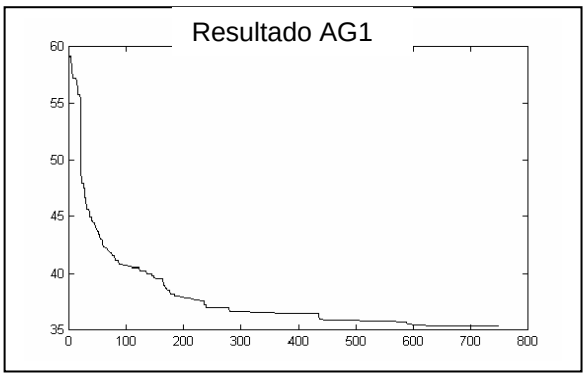
O programa de montagem e mapa de alimentação foram transferidos do HLC para a máquina JUKI e procedeu-se à montagem em modo ‘*Dry Ram*’. Nesse modo, a máquina executa todas os movimentos de montagem sem no entanto precisar estar alimentada com componentes.

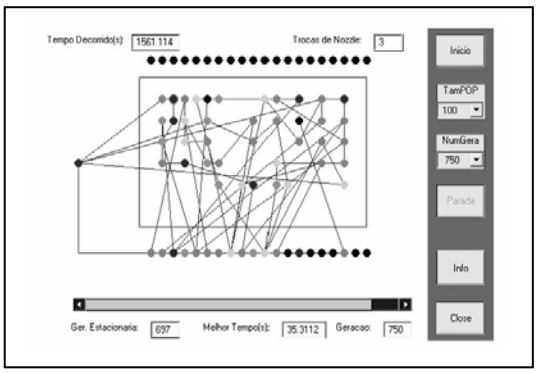
A **solução** final obtida pelo **HLC** para o problema exemplo, em termos de tempo total de montagem **cronometrado em máquina**, foi de **28,2s por placa**.

Nas próxima seção, serão apresentados as soluções propostas pelo AG. Para melhor compreensão dos resultados, utilizar-se-á a mesmo formato de representação visual adotado na tabela 4.3.

4.4. Resultados do AG

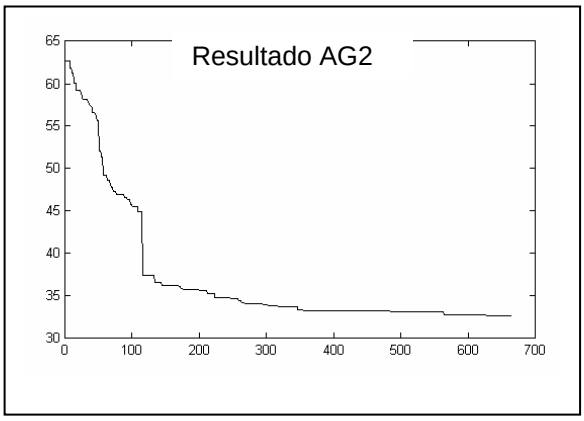
Com o mesmo conjunto de dados do problema exemplo, o AG foi executado 5 vezes. As soluções encontradas estão tabuladas a seguir em conjunto com a **curva de convergência** do AG e a **plotagem da trajetória** do braço (ou ‘caminho de rato’, conforme o jargão utilizado pelos técnicos da área), ambas implementados na interface do AG utilizando os recursos gráficos do Matlab:

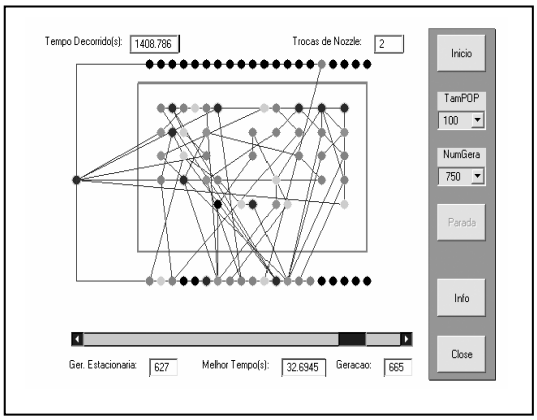




Ciclo	H1	H2	H3	H4	Posições				Feeders				Head placement sequence			
1	2	2	3	4	b3	b4	c3	l4	03	03	04	05	4	2	3	1
2	2	2	3	4	b6	b1	c5	m1	03	03	04	02	4	1	2	3
3	2	2	3	4	b2	g1	d1	e4	03	14	10	09	1	2	4	3
4	2	2	3	4	b7	b5	c2	l3	03	03	04	05	3	1	2	4
5	4	4	3	1	l2	j2	d4	f1	05	06	10	11	3	4	1	2
6	4	4	3	1	e3	e1	d3	f2	09	09	10	11	3	4	2	1
7	4	4	3	1	l1	m3	c4	a1	05	02	04	08	1	3	4	2
8	4	4	3	1	i1	m2	c1	a2	01	02	04	08	2	1	3	4
9	4	4	3	1	j1	e2	d2	f4	06	09	10	11	4	2	1	3
10	4	4		1	j3	k1		a3	06	07		08	3	1		2
11	4			1	n2			a4	12			08	2			1
12	4			1	n1			f3	12			11	1			2
13	4				h1				18				1			

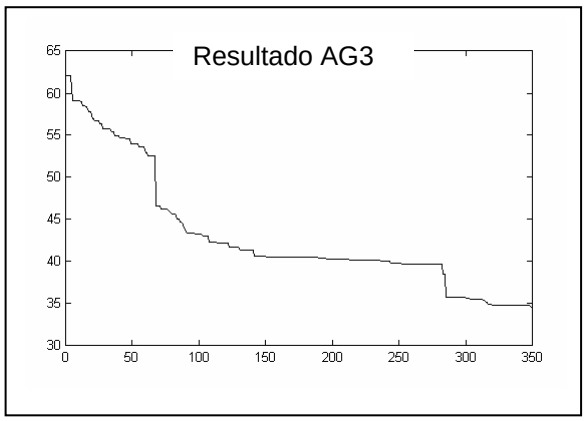
Figura 4.7 – Solução No.1 do AG para o exemplo da tabela 5.1

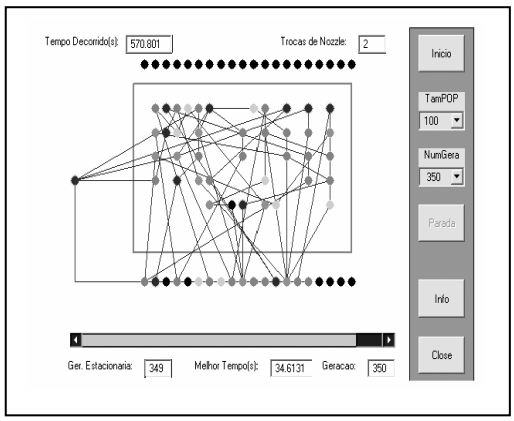




Ciclo	H1	H2	H3	H4	Posições				Feeders				Head placement sequence			
1	4	4	1	3	j1	m2	f4	d3	09	01	02	07	4	1	2	3
2	4	4	1	3	k1	l2	f1	d4	08	03	02	07	4	3	1	2
3	4	4	1	3	h1	e1	a3	c2	14	15	11	13	4	3	2	1
4	4	4	1	3	l4	m3	f2	d1	03	01	02	07	1	3	2	4
5	4	4	2	3	e3	e2	b3	c1	15	15	12	13	1	3	2	4
6	4	4	2	3	j2	n1	b1	c3	09	10	12	13	2	1	3	4
7	4	4	2	3	j3	n2	g1	d2	09	10	06	07	3	4	1	2
8	4	1	2	3	i1	a4	b2	c4	36	11	12	13	3	2	1	4
9	4	1	2	3	m1	f3	b6	c5	01	02	12	13	1	3	4	2
10	4	1	2		e4	a1	b7		15	11	12		2	3	1	
11	4	1	2		l1	a2	b5		03	11	12		3	2	1	
12	4		2		l3		b4		03		12		1		2	

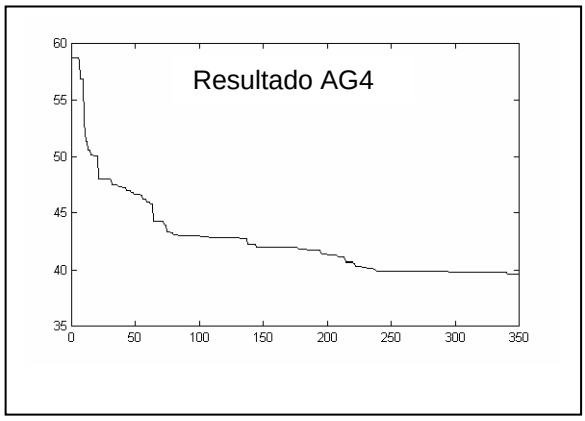
Figura 4.8 – Solução No.2 do AG para o exemplo da tabela 5.1

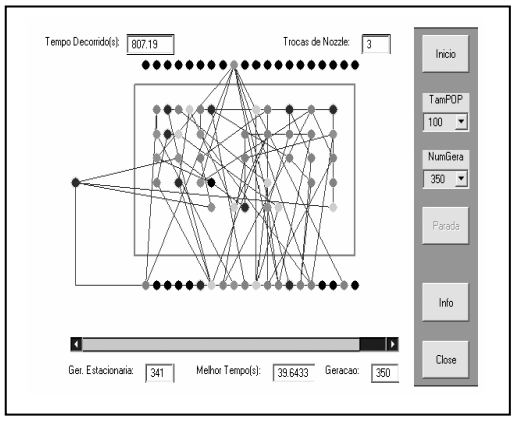




Ciclo	H1	H2	H3	H4	Posições	Feeders	Head placement sequence
1	4	1	1	3	e1 f1 a4 d3	04 06 08 10	3 4 2 1
2	4	1	1	3	j2 f3 f4 c2	09 06 06 14	2 1 3 4
3	4	1	1	3	l3 a2 f2 d4	01 08 06 10	4 3 2 1
4	4	1	1	3	h1 a1 a3 c3	07 08 08 14	1 3 4 2
5	4	4	2	3	j3 l2 b2 d1	09 01 13 10	3 2 1 4
6	4	4	2	3	e3 e2 b1 c5	04 04 13 14	2 1 4 3
7	4	4	2	3	k1 m1 b6 c4	15 11 13 14	2 1 3 4
8	4	4	2	3	j1 m2 b4 d2	09 11 13 10	4 3 2 1
9	4	4	2	3	e4 m3 b7 c1	04 11 13 14	4 2 3 1
10	4	4	2		n2 n1 b5	12 12 13	3 2 1
11	4	4	2		l1 l4 g1	01 01 02	3 1 2
12		4	2		i1 b3	16 13	2 1

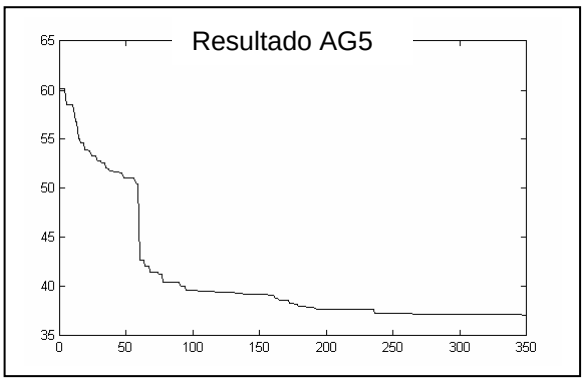
Figura 4.9 – Solução No.3 do AG para o exemplo da tabela 5.1

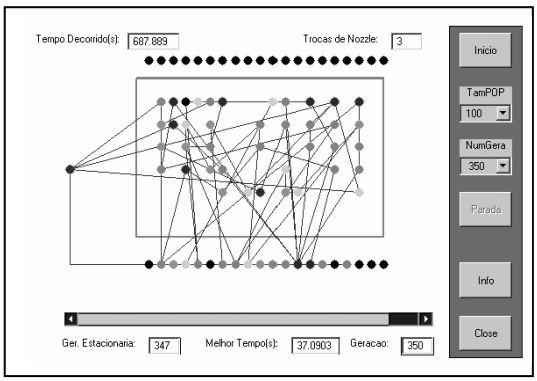




Ciclo	H1	H2	H3	H4	Posições	Feeders	Head placement sequence
1	1	4	2	1	a1 m3 b5 f1	11 13 14 07	1 3 2 4
2	1	4	2	1	a2 e4 b7 f4	11 12 14 07	2 3 1 4
3	1	4	2	1	a4 e2 g1 f2	11 12 06 07	3 4 1 2
4	1	4	2	1	f3 l2 b1 a3	07 08 14 11	4 3 2 1
5	4	3	2	4	l4 d2 b6 e3	08 09 14 12	4 3 1 2
6	4	3	2	4	j3 c3 b4 n2	16 29 14 15	4 1 2 3
7	4	3	2	4	h1 d4 b3 n1	10 09 14 15	4 3 1 2
8	4	3	2	4	m2 c4 b2 m1	13 29 14 13	4 1 3 2
9	4	3		4	j2 c5 j1	16 29 16	1 2 3
10	4	3		4	l1 d3 e1	08 09 12	3 1 2
11	4	3			k1 c1	19 29	2 1
12	4	3			l3 d1	08 09	2 1
13	4	3			i1 c2	01 29	2 1

Figura 4.10 – Solução No.4 do AG para o exemplo da tabela 5.1





Ciclo	H1	H2	H3	H4	Posições	Feeders	Head placement sequence
1	1	3	1	4	f2 c3 a3 m3	09 03 04 05	4 1 2 3
2	1	3	1	4	a1 c1 a2 m2	04 03 04 05	1 3 2 4
3	1	3	1	4	f4 d4 f1 j2	09 10 09 08	3 2 1 4
4	1	3	1	4	f3 d2 a4 k1	09 10 04 07	2 3 4 1
5	4	3	4	2	l4 d1 n1 b3	02 10 11 13	3 4 1 2
6	4	3	4	2	l1 c4 e1 b4	02 03 12 13	3 4 1 2
7	4	3	4	2	j1 d3 n2 g1	08 10 11 14	3 1 2 4
8	4	3	4	2	l2 c5 e3 b1	02 03 12 13	1 3 2 4
9	4	3	4	2	l3 c2 e2 b2	02 03 12 13	3 4 2 1
10	4		4	2	m1 e4 b5	05 12 13	3 2 1
11	4		4	2	j3 h1 b6	08 17 13	1 3 2
12	4			2	i1 b7	15 13	1 2

Figura 4.11 – Solução No.5 do AG para o exemplo da tabela 4.1

A melhor **solução** obtida pelo **AG** para o problema exemplo, em termos de tempo total de montagem **cronometrado em máquina**, foi de **29,3s por placa (correspondente ao resultado AG5)**.

Na próxima seção, será feita a análise qualitativa e quantitativa comparando as soluções apresentadas aqui com base nos tempos reais de máquina. Vale lembrar que os valores em tempo obtidos pela máquina virtual nas figuras 4.7 a 4.11 não servem como referência para comparação entre soluções do AG, pois alguns pesos da função objetivo foram alterados entre uma otimização e outra na tentativa de encontrar o melhor ponto.

4.6. Análise Comparativa das Soluções

As soluções, após a otimização, obtidas pelo AG foram transferidas, uma a uma, para a máquina JUKI KE-2030 para serem executadas em modo '*Dry Ram*'. Isso permitiu a comparação de resultados em uma mesma base para o HLC e AG. Na tabela 4.4 encontram-se os resultados qualitativos e quantitativos da solução encontrada pelo HLC contra as 5 soluções obtidas com o AG.

Tabela 4.4 – Análise comparativa das soluções propostas pelo AG e HC.

	Análise comparativa de parâmetros:										
	Qualitativos									Quantitativos	
No#	**Ciclos**	**TN**	**ANC**	**D-H**	**N-F**	**PS4**	**PS3**	**PS2**	**PNS**	**Tempo(s)**	**Melhoria**
HLC	**14**	**4**	1	**6**	**17**	**4**	4	5	**1**	28,2	-
AG#1	13	3	1	4	14	0	4	5	4	30,4	-7,80%
AG#2	12	2	2	3	14	0	1	9	2	32,6	-15,60%
AG#3	12	2	1	3	14	0	0	6	6	31,1	-10,28%
AG#4	13	3	1	5	14	0	0	10	3	31,3	-10,99%
AG#5	12	3	1	3	14	0	2	8	2	29,3	-3,90%

Abaixo, o significado das siglas utilizadas na tabela 4.4:

Ciclos : número total de ciclos de *pick & place* para a montagem da placa.
TN : total de trocas de *nozzle*
ANC : total de visitas ao ANC
D-H : desbalanceamento de carga de trabalho entre os cabeçotes
N-F: : quantidade de alimentadores utilizados nos magazines
PS4: : número de ciclos com *pickup* simultâneo usando 4 cabeçotes
PS3: : número de ciclos com *pickup* simultâneo usando 3 cabeçotes
PS2: : número de ciclos com *pickup* simultâneo usando 2 cabeçotes
PNS: : número de ciclos sem nenhum *pickup* simultâneo
Tempo(s): : tempo real cronometrado para a montagem dos componentes na placa.
Melhoria: : percentual de melhoria em relação à solução do HLC, calculado como

(Tempo HLC – Tempo AG)/Tempo HLC (%)

A média dos tempos de montagem cronometrados na máquina JUKI em modo '*Dry Ram*' com as soluções do AG, para o problema exemplo, ficou em **30,9s** com desvio padrão de **1,21s**. A relação desse desvio padrão relação à média ($\sigma/\overline{x}$) ficou em **3,92%,** que denominou-se de índice de variação das soluções.

O melhor tempo de montagem obtido com o AG, dentre as cinco soluções cronometradas, ficou **3,9%** acima do tempo de montagem obtido com o HLC.

Capítulo 5

Conclusões e Pesquisa Futura

O objetivo inicial desse trabalho era propor uma solução, utilizando a técnica de algoritmos genéticos, para o complexo problema de otimização da montagem de componentes eletrônicos em placas de circuito impresso utilizando máquinas *pick&place* do tipo *Multi-Head*.

Durante a fase de estudo da literatura existente sobre esse tipo de problema, encontrou-se publicações acerca dos diversos tipos de máquinas SMT, tais como: *turret head*, *rotary head (pick&place)* e *multi-head pick&place*. Esse último tipo foi escolhido, além de todos os outros motivos citados na introdução, devido à disponibilidade para a realização de testes práticos em ambiente industrial.

Após a escolha da máquina e seleção dos principais trabalhos relacionados [5a, 7, 15, 18], estudou-se em detalhes a abordagem proposta por Lee et al. [5] haja vista a utilização de algoritmos genéticos e a riqueza de informações sobre sua implementação no material publicado. Estudou-se, ainda, os manuais técnicos das máquinas JUKI KE-2030 a fim de conhecer os recursos e especificações de tempo do equipamento. Logo verificou-se que as restrições impostas por Lee et al. [5] acerca da ordem fixa de acionamento dos cabeçotes e da impossibilidade de repetição do mesmo tipo de nozzle em mais de um cabeçote levariam a soluções com desempenhos inferiores, pois alguns recursos vitais para o máximo aproveitamento e eficiência da máquina não estariam sendo utilizados.

Surgiu, daí, a decisão de desenvolver e implementar na prática um trabalho que rompesse com as restrições impostas nos trabalhos existentes e trouxesse uma contribuição à literatura e pesquisas sobre o tema. O software HLC precisou ser utilizado como parâmetro de comparação para a validação do AG proposto nesse

trabalho, tendo em vista a indisponibilidade de resultados na literatura equiparáveis a esse tipo versátil de máquina MH.

A ferramenta de *software* utilizada para a programação e execução do algoritmo, Matlab, atendeu ao propósito da implementação. No início do trabalho, os códigos fontes foram desenvolvidos da forma convencional com a utilização de *loops* *'for'* para varrer todos os cromossomos da população, um a um, durante às chamadas às funções de operação genéticas: cruzamento, mutação, função objetivo, etc. Como exemplo, para rodar 500 gerações de uma população de 100 cromossomos de tamanho 50, seriam necessárias 2,5 milhões de chamadas a função de cálculo do tempo de deslocamento entre duas cidades (posições na placa). Para contornar o problema e viabilizar a execução do algoritmo com os recursos computacionais disponíveis, seção 1.4, o código fonte foi remodelado utilizando a poderosa técnica de vetorização do Matlab, que elimina a necessidade de uso de *loops 'for'*, otimizando significativamente o tempo de execução do programa.

Os operadores genéticos desenvolvidos para mutação, cruzamento e seleção foram apropriados e apresentaram resultados satisfatórios em relação às características de convergência do algoritmo. Destaca-se, aqui, o operador de mutação heurística NSMλ [19] de fundamental importância para manter a diversidade da população ao longo das gerações e encontrar saídas para as regiões de platôs (ótimos locais), embora com um custo elevado, comparado aos demais, em relação ao tempo de execução do AG.

A técnica de componentes vazios descrita na seção 3.2.5 em conjunto com o desacoplamento do problema de TAP, por meio da criação do *LinkN*, possibilitou tratar o caso MH-NR de forma eficiente, desonerando o AG principal. Isso não só resultou na redução de tempo do AG, como permitiu encontrar soluções para o arranjo de nozzles nos cabeçotes que estavam fora do espaço de busca da abordagem inicial para o caso MH.

Em relação aos resultados práticos obtidos com os experimentos realizados, pode-se concluir que o AG proposto nesse trabalho é eficiente para resolver os problemas de TAP, PAPSP e RAP, face à hipótese assumida sobre o não uso de mais de um carretel do mesmo tipo de componente. Tal hipótese limita o espaço de busca, impossibilitando soluções que maximizem o *pickup* simultâneo, o que não acontece no HLC.

Em conseqüência disso, a análise quantitativa dos resultados mostrou que o tempo de montagem correspondente a melhor solução obtida com o AG fica em torno de **3,9%** acima do tempo de montagem obtido com o HLC. A partir desse resultado conclui-se que existem oportunidades de melhoria do AG proposto nesse trabalho.

Pesquisa Futura

Os resultados obtidos com a implementação prática desse trabalho revelam pontos que podem ser melhorados no AG a fim de propiciar soluções no mesmo nível, ou até melhores, do que o algoritmo comercial HLC utilizado para comparação. São eles:

(1) Aumentar o número de ciclos com PS4 (*pickup* simultâneo utilizando os 4 cabeçotes).

Proposta: não considerar a hipótese restritiva do uso de apenas um carretel por tipo de componente nos slots do magazine, aumentando o espaço de busca para o problema de RAP, pois um dos fatores principais para obter o melhor desempenho da máquina *Multi-Head* estudada é maximar o pickup simultâneo dos componentes nos alimentadores. Para essa abordagem, que também não foi encontrada na literatura, propõe-se a nomenclatura MH-NR/FR (*Multi-Head* com *nozzles* repetidos e *feeders* repetidos);

(2) Índice de variação das soluções alto para aplicações reais (3,92%), medido a partir dos tempos cronometrados das soluções obtidas com o AG executado árias vezes para um mesmo problema. Essa variação exigiu que cada experimento fosse rodado, no mínimo, 5 vezes para obter uma solução aceitável.

Proposta: Investigar o efeito da variação dos parâmetros do AG (percentuais de mutação, cruzamento, pesos da função objetivo, métodos de seleção, etc.) no comportamento da curva de convergência de forma a reduzir o desvio padrão das soluções obtidas. Como a análise envolve vários fatores que afetam um único resultado, sugere-se para esse propósito a utilização da técnica estatística denominada Projeto de Experimentos;

(3) Tempo de execução do AG alto para aplicações reais.

Proposta: como foi mostrado na figura 4.2, a função objetivo e mutação heurística são juntas responsáveis por 90,7% do tempo total do AG. Para melhorar esse resultado sugere-se:

- reduzir a complexidade da função objetivo para as primeiras 300 gerações, avaliando-se a qualidade das soluções, durante esse período, por meio de uma função simplificada que considere somente os principais parâmetros que afetam o resultado final (tempo de montagem)
- substituir o operador de mutação heurística por outro que onere menos o tempo de processamento
- utilizar o conceito de paralelização do AG.

Referências

[1] TIRPAK, T. M. “Simulation Software for Surface Mount Assembly”. In: *Proceedings of the 1993 Winter Simulation Conference*, pp.796-803, Los Angeles, CA, 1993.

[2] AYOB, M., COWLING, P., KENDALL, G. “Optimization for Surface Mount Placement Machines”, *IEEE International Conference on Industrial Technology*, pp.486-491, 2002.

[3] KUMAR, R. AND LI, H. “Integer Programming Approach to Printed Circuit Board Assembly Time Optimization”, *IEEE Trans. Components, Packaging and Manufacturing Technology - Part B,* v. 18, n. 4, pp.720-727, Nov. 1995.

[4] LEIPALA, T., NEVALAINEN, O. “Optimization of the Movements of a Component Placement Machine”, *European Journal of Operational Research*, v. 38, pp.167-177, 1989.

[5] LEE, W., LEE, S., LEE, B., et al. “A Genetic Optimization Approach to Operation of a Multi-head Surface Mounting Machine”, *IEICE Trans. Fundamentals*, v. E83-A, n. 9, pp. 1748-1756, Sep. 2000.

[6] TIRPAK, T. M., NELSON, P.C., ASWANI, A.J., “Optimization of Revolver Head SMT Machines using Adaptive Simulated Annealing (ASA)”. In: *Proceedings of the International Electronics Manufacturing Technology Symposium*, 2000 IEEE/CPMT, pp.214-220, Santa Clara, CA, 2000.

[7] BURKE, E. K., COWLING, P. I., KEUTHEN, R. “New Models and Heuristics for Component Placement in Printed Circuit Board Assembly”. In: *Proceedings of the 1999 International Conference on Information Intelligence and Systems*, pp.133-140, 1999.

[8] RODRIGUES, R. *Complexidade Computacional – NP Completude*. ICE –

Departamento de Ciência da Computação, UFAM, 2000. http://www.dcc.ufam.edu.br/~rosiane/cursos/grad/paa0201/comple~1.ppt

[9] BURKE, E. K., COWLING, P. I., KEUTHEN, R. "Effective Heuristics and Metaheuristics Approaches to Optimize Component Placement in Printed Circuit Board Assembly". In: *Proceedings of the 2000 Congress on Evolutionary Computation*, v.1, pp.301-308, 2000.

[10] BURKE, E. K., COWLING, P. I., KEUTHEN, R., "The Printed Circuit Board Assembly Problem: Heuristics Approach for Multi-Head Placement Machinery", *International Conference on Artificial Intelligence*. 2001. http://www.asap.cs.nott.ac.uk/publications/ps/ralf_PCBAP.ps

[11] KEUTHEN, R. *Heuristic Approach for Routing Optimization*, Ph.D. thesis, University of Nottingham, UK, Jan. 2003. http://www.asap.cs.nott.ac.uk/publications/theses.shtml

[12] AYOB, M., KENDALL, G. "Real-time scheduling for multi headed placement machine". In: *Proceedings of the IEEE International Symposium on Assembly and Task Planning*, pp.128-133, Besançon, France, 2003.

[13] BALL, M., MAGAZINE, M. "Sequencing of Insertion in Printed Circuit Board Assembly", *Operations Research*, v. 36, n. 2, pp.192-210, 1989.

[14] HO, W., JI, P., "A Hybrid Genetic Algorithm for Component Sequencing and Feeder Arrangement", *Journal of Intelligent Algorithm*, v.15, pp.307-315, 2004.

[15] JEEVAN, A. P., SEETHARAMU, K. N., AZID, I. A., et al. "Optimization of PCB Component Placement Using Genetic Algorithms", *Journal of Electronics Manufacturing*, v. 11, n. 1, pp.69-70, 2002.

[16] GEN, M., CHENG, R., *Genetic Algorithms & Engineering Design*. 1 ed. New

York, John Wiley & Sons, Inc., 1997.

[17] MICHALEWICZ, Z., *Genetic Algorithms + Data Structures = Evolution Programs.* 2 ed. Berlin Heidelberg, Springer-Verlag, 1994.

[18] CHIPPERFIELD, A., FLEMING, P., POHLHEIM, H., et al. *Genetic Algorithm Toolbox – User's Guide vs.1.2,* Department of Automatic Control and Systems Engineering, University of Sheffield. http://www.shef.ac.uk/~gaipp/ga-toolbox/

[19] TSUJIMURA, Y., CHENG R., GEN, M. "Improved Genetic Algorithms for Job-Shop Scheduling Problems", *Engineering Design & Automation 3(2)*, v. 3, n. 2, pp.133-144, 1997.

[20] DEB, KALYANMOY, *Multi-Objective Optimization using Evolutionary Algorithms.* 1 ed. Chichester, John Wiley & Sons, Ltd., 2001.

[21] MITCHELL, M., *An Introduction to Genetic Algorithms*. 1 ed. Massachusetts, MIT Press, 1996.

[22] MAN K.F., TANG K.S., KWONG, S., *Genetic Algorithms*. 1 ed. London, Springer-Verlag, 1999.

[23] DAVIS, L. "Job Shop Scheduling with Genetic Algorithms". In: *Proc. 1st Int. Conf Genetic Algorithms*, pp.136-140, 1985.

[24] BOOKER, L. "Improving Search in Genetic Algorithms", *Genetic Algorithms and Simulated Annealing*, pp.61-73, 1987.

[25] DAVIS, L. "Adapting operator probabilities in genetic algorithms". In: *Proc. 3rd. Int. Conf. Genetic Algorithms*, pp. 61-79, 1991.

[26] DAVIS, L., *Handbook of Genetic Algorithms.* 2 ed., New York, Van Nostrand Reinhold, 1991.

[27] BAKER, J. “Adaptive Selection Methods for genetic algorithms”. In: *Proceedings of the Second International Conference on Genetic Algorithms*, VOLUME, pp. 100-111, Hillsdale, NJ, 1987.

Apêndice

O que é um AG ?

Algoritmo Genético (AG) é um método estocástico de busca global que se inspira no processo biológico conhecido como evolução. O AG atua sobre um conjunto de soluções potenciais, denominado de população, por meio de seus operadores de cruzamento, mutação e seleção, impondo o princípio de sobrevivência dos indivíduos mais aptos (*survival of the fittest*) a fim de produzir soluções melhores a cada geração. O genótipo de um indivíduo é codificado em uma estrutura numérica chamada de cromossomo, unicamente mapeado ao seu fenótipo (variável de decisão) por meio de uma Função Objetivo. Tal função exerce o papel do ambiente de adaptação dos indivíduos e traduz o grau de aptidão de um cromossomo candidato à solução.

De uma forma simplificada, um AG pode ser caracterizado pelo diagrama em blocos na figura A.1 [16], que ilustra o ciclo de evolução de uma população e demonstra onde ocorrem os processo de codificação das soluções em cromossomos e decodificação dos cromossomos em soluções.

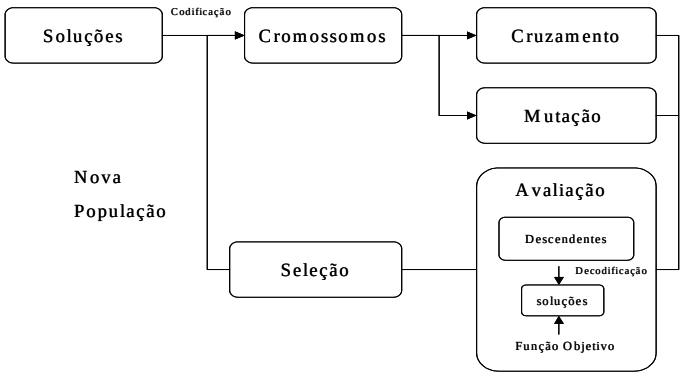


Figura A.1 – Estrutura Geral dos Algoritmos Genéticos

Soluções do TSP utilizando AGs (Metaheurística)

Os AGs, inicialmente, foram desenvolvidos com a finalidade de resolver problemas combinatórios considerados difíceis e cuja solução, por meio dos métodos analíticos convencionais, torna-se computacionalmente inviável. Uma única e simples representação dos algoritmos genéticos, na forma de estrutura de cromossomos e operadores genéticos, não seria adequada para resolver os complexos e diversos problemas de otimização. Assim, diferentes implementações dos AGs, utilizando formas distintas de representação dos cromossomos (binária, inteira, real, etc.) foram desenvolvidas e utilizadas como metaheurísticas.

Procura-se encontrar a forma mais natural possível de representar cada problema, individualmente, e criar os operadores genéticos que melhor se adeqüem à estrutura de dados utilizada [21]. Nesse trabalho, cujo problema assemelha-se ao clássico Caixeiro Viajante (*TSP – Traveling Salesman Problem*), a forma natural encontrada para representar os cromossomos é a lista ordenada ou seqüência. Os operadores genéticos (cruzamento, mutação e seleção) mais indicados para esse tipo de estrutura serão descritos, em detalhes, nos tópicos a seguir.

Vale ressaltar as principais vantagens dos AGs quando comparados às demais técnicas de otimização [16]:

- AGs não requerem a síntese de equações matemáticas para solucionar um problema. Buscam as soluções ótimas sem, necessariamente, conhecer as características intrínsecas do problema.
- Eficiência para efetuar a busca do ótimo global. Os AGs não exigem conhecimento sobre as propriedades de convexidade ou não do problema. Naturalmente, exploram todo o espaço de busca.
- Flexibilidade para adicionar heurísticas na implementação do algoritmo, que visem melhorar o desempenho do AG para um problema específico.

Operadores Genéticos

Conforme diagrama da figura A.1 as soluções do problema real são codificadas em estruturas denominadas ‘cromossomos’. Cada cromossomo representa um indivíduo de uma população. A evolução da população depende de como os cromossomos

(indivíduos) vão interagir entre si e serem escolhidos para formar as gerações futuras. A ‘interação’ de cromossomos é realizada por meio de operadores genéticos de cruzamento e mutação, enquanto a ‘escolha’ é realizada por meio de um processo de avaliação e seleção.

Cruzamento

Também denominado de reprodução, o cruzamento pressupõe a participação de dois cromossomos para dar origem a um ou mais descendentes. O intercâmbio de genes pode dar-se de diversas formas possíveis. Em AGs, procura-se encontrar as técnicas de cruzamento mais eficientes para cada problema, especificamente, de acordo com sua natureza.

A idéia por trás do operador de cruzamento (*crossover*) é promover a geração de indivíduos a partir de estruturas genéticas de seus antecessores, aproveitando o que há de melhor em cada um deles. Com isso, ao longo do ciclo evolutivo, as novas gerações tendem a ser compostas por descendentes com maiores chances de sobrevivência ou, em outras palavras, maior grau de aptidão ao meio.

Conceitualmente, o cruzamento apresenta uma característica desfavorável, a longo prazo, na busca do ótimo global, por ser um operador que torna a população mais homogênea: tende a contribuir para a formação de blocos de construção que, após algumas gerações, terá produzido uma população de indivíduos semelhantes, que habitam em uma região do espaço de busca onde existe um ótimo local, não necessariamente global. Para contornar o problema, faz-se necessário incluir, no AG, um operador que torne a população mais heterogênea: a mutação, que será detalhada posteriormente.

Para o clássico problema do Caixeiro Viajante, em que os cromossomos são, normalmente, representados por seqüências de números inteiros (cada um desses números é um gene que representa uma cidade a ser visitada), é necessário desenvolver técnicas específicas que preservem a validade da seqüência gerada após o cruzamento, pois nenhuma cidade poderá deixar de ser visitada, ou ser visitada mais de uma vez.

Técnicas de Cruzamento

Dentre as diversas técnicas de cruzamento aplicáveis, destacam-se:

- Cruzamento Ordenado (*OX – Ordered Crossover*):
- Cruzamento por Mapeamento Parcial (*PMX – Partial-Mapped Crossover*):
- Cruzamento Baseado em Posição (*Position Based Crossover*):
- Cruzamento Baseado em Ordem (*Ordered Based Crossover*):
- Cruzamento Heurístico

Mutação

Durante o ciclo evolutivo, as gerações dão origem à novas gerações por meio do cruzamento e reprodução de seus indivíduos, como visto na seção anterior. Alguns descendentes, além da herança genética dos seus ancestrais, trazem uma característica nova resultante de um processo de mutação de alguns genes de seu cromossomo. Biologicamente, tais mutações são erros de cópia de alguns nucleotídeos que formam o DNA.

Em AG, o processo de mutação tem papel fundamental na diversificação dos indivíduos de uma população. Durante a evolução, pode ocorrer a estabilização da curva de adaptação dos indivíduos em um platô que representa uma região de ótimo local. A mutação exerce, então, a função de deslocar alguns indivíduos da população para outra região do espaço de busca, dando prosseguimento ao ciclo evolutivo em direção à região em que se encontra o ótimo global.

Nesse aspecto, é essencial caracterizar a diferença entre uma busca genética e uma busca randômica. Essa ocorre, normalmente, na chamada programação evolucionária, em que tão somente há a mutação dos indivíduos de uma população, não havendo o processo de cruzamento e troca de informação genética entre os indivíduos. Já na busca genética, que ocorre nos AGs, há tanto o processo de mutação quanto o de cruzamento, sendo que a evolução é, fundamentalmente, uma conseqüência da troca de material genético entre os indivíduos de uma geração para dar origem a seus descendentes que, por sua vez, podem ser melhores por possuírem um maior grau de adaptação ao meio.

Isso sugere, numa primeira análise da teoria dos AGs, uma relação prática entre as probabilidades de cruzamento (*pc*) e de mutação (*pm*) dos indíviduos de uma população. É de se esperar que *pc* seja muito maior do que *pm* (*pc* >> *pm*), a fim de manter as características de uma busca genética citada anteriormente. No entanto, há controvérsias sobre a escolha dos índices de probabilidade de mutação e cruzamento, pois o aumento de *pc* pode levar à formação de blocos de construção (*building blocks*) e, ao mesmo tempo, contribuir para a destruição de bons cromossomos. Por outro lado, o aumento de *pm* contribui para reintroduzir o material genético perdido, embora tenda a transformar a busca em randômica [22].

Alguns métodos são sugeridos na literatura para contornar esse dilema. Davis [23] propôs a variação linear decrescente de *pc* e crescente de *pm* no decorrer do ciclo evolutivo. Isso, em outras palavras, significa reintroduzir material genético na população, com maior freqüência, à medida em que os indivíduos tendem a se tornar semelhantes após sucessivos processos de cruzamento e seleção. Booker [24] utilizou um índice de *pc* dinâmico, dependente da variação do grau de aptidão entre os indivíduos de uma população. Davis [25,26] sugeriu, ainda, a modificação dos parâmetros *pc* e *pm*, durante a evolução, de acordo com o sucesso em gerar melhores indivíduos descendentes. Man, Tang e Kwong [22] recomendam, como ponto de partida e regra prática, a utilização das seguintes probabilidades de cruzamento e mutação: pc = 0,60 e pm = 0,001 para população grande (~100) e pc = 0.9 e pm = 0.01 para população pequena (~30) .

Técnicas de Mutação

As técnicas convencionais de mutação voltadas ao problema do Caixeiro Viajante, em que os cromossomos são, normalmente, representados por listas ordenadas (*ordered lists*) ou seqüência de números inteiros, são:

- Inversão (*invertion Mutation*): escolhem-se duas posições aleatórias de um cromossomo e, então, inverte-se o bloco (ou partição) de genes compreendido entre essas.
- Inserção (*Insertion Mutation*): escolhe-se uma posição aleatória do cromossomo a ser inserida em uma outra posição aleatória.

- Deslocamento (*Displacement Mutation*): uma partição do cromossomo é aleatoriamente escolhida e inserida em outra posição do cromossomo.
- Troca Recíproca (*Reciprocal Exchange Mutation*): duas posições do cromossomo são escolhidas, aleatoriamente, e intercambiadas.
- Heurística (*Heuristic Mutation*): essa técnica, proposta por Gen et al. [16], utiliza o conceito de vizinhança (neighborhood) para que a mutação produza um indivíduo melhor. Adiante, na descrição do AG implementado nesse trabalho, será fornecida uma descrição mais detalhada do assunto, denominada NSMλ [19].

Função Objetivo e Função de Aptidão

Biologicamente, alguns indivíduos de uma população são mais aptos à sobreviver em um determinado ambiente de desafio do que outros de sua mesma espécie. Para dintingüi-los apropriadamente, um valor deve ser atribuído a cada indivíduo qualificando-o em relação a esse desafio ou **objetivo** em questão.

Uma **Função Objetivo** *f* é a forma utilizada nos algoritmos genéticos de realizar essa medição e obter o status (valor objetivo) de cada indivíduo ou cromossomo [22]. Pode-se dizer que a função objetivo é a interface entre o AG e o sistema em estudo, pois é ela que será responsável pela decodificação do cromossomo e avaliação de mesmo no espaço da variável que está sendo otimizada.

Haja vista que todos os cromossomos de uma população passam pela mesma forma de avaliação, a magnitude dos valores obtidos variam de um cromossomo para outro. A fim de manter a uniformidade, os valores objetivos *O* são mapeados em uma novo domínio *F* por uma transformação $g : O \rightarrow F$.

Define-se a **Função de Aptidão** (*Fitness*) como sendo: $F(x) = g(f(x))$, em que o valor de $F(x)$ é chamado de **grau de aptidão relativo**, ou simplesmente, **aptidão**.

Algumas técnicas citadas na literatura [22,18] para promover a transformação *g* de *f*(*x*) em *F*(*x*) são:

- Escalonamento Linear (*Linear Scaling*): o relacionamento entre as duas funções é linear:

$$F(x) = af(x) + b \qquad \text{(A.1)}$$

- Aptidão Proporcional (*Proportional Fitness Assignment*): a aptidão de um indivíduo é calculada como a relação entre o valor da função objetivo desse indivíduo e da soma dos valores da função objetivo de todos os indivíduos da população:

$$F(x_i) = \frac{f(x_i)}{\sum_{i=1}^{n} f(x_i)} \tag{A.2}$$

- Pontuação (*Ranking*): essa técnica consiste em atribuir graus de aptidão aos indivíduos, dentro de um intervalo, tipicamente, [0, 2]. Antes de aplicar a equação A.3 abaixo, é necessário realizar uma classificação (*sort*) dos indivíduos pelo valor objetivo em ordem decrescente, do menos apto ao mais apto, tratando-se de um problema de minimização. A variável *MAX*, normalmente escolhida no intervalo [1.1, 2.0], define o *bias* (tamanho do intervalo) ou pressão seletiva:

$$F(x_i) = 2 - MAX + 2(MAX - 1)\frac{x_i - 1}{n - 1} \tag{A.3}$$

Uma das conseqüências graves de realizar o processo de seleção diretamente sobre os valores objetivos é a convergência prematura. A explicação para esse fato é que aqueles indivíduos extremamente aptos irão, em poucas gerações, dominar o processo de reprodução causando rápida convergência para possíveis ótimos locais.

O uso de técnicas como a Pontuação, descrita acima, contribui significativamente para o desempenho de convergência do algoritmo, embora possua um ‘custo’ adicional decorrente do tempo de classificação dos cromossomos.

Seleção

No decorrer do processo evolutivo, na transição de uma geração para outra, os indivíduos de uma população passam por um processo de seleção natural que escolhe quem são os mais aptos a reproduzir os descendentes da geração seguinte. Com isso, o propósito da seleção é identificar os indivíduos da população com maior grau de aptidão na esperança de que seus descendentes sejam ainda melhores, ou seja, possuam graus de aptidão mais elevados do que seus pais.

A severidade do processo de seleção precisa ser bem balanceada: um processo muito rígido implicará em que indivíduos com características sub-ótimas tomarão, rapidamente, o espaço de toda a população, reduzindo a diversidade necessária para a evolução das gerações futuras. Por outro lado, um processo de seleção muito fraco resultará em um ciclo evolutivo muito lento.

Man et al. [22] apresentam três indicadores de desempenho de algoritmos de seleção: *bias*, *spread* e eficiência.

Define-se *bias* como sendo a diferença absoluta entre o **valor real** (i.e., a quantidade de indivíduos selecionados de fato) e o **valor esperado** (i.e., o número esperado de vezes que um indivíduo será selecionado para reproduzir). O valor ótimo de *bias* é zero, o que significa que a probabilidade real de seleção dos indivíduos corresponde à probabilidade esperada.

Spread corresponde à variação (*range*) do valor esperado de um indivíduo. Se $f(i)$ é o valor real de um determinado indivíduo i, então o "*spread* mínimo" é o menor valor de *spread* que teoricamente permite *bias* zero, isto é: $f(i) \in \{\lfloor et(i) \rfloor, \lceil et(i) \rceil\}$, onde *et*(*i*) corresponde ao valor esperado de um indivíduo i, sendo $\lfloor et(i) \rfloor$ o mínimo e $\lceil et(i) \rceil$ o máximo.

Sendo assim, enquanto o *bias* é um indicador de acuracidade, o *spread* de um processo de seleção é uma forma de medir sua consistência. O indicador de eficiência está relacionado à complexidade de tempo do algoritmo de seleção, visando não sobrecarregar a complexidade de tempo de todo o AG.

Método da Roleta

Diversas técnicas de seleção fazem uso da **roleta** a fim de selecionar indivíduos de uma população de forma probabilística baseando-se em alguma medida de seu desempenho.

Um intervalo de valor real, *Sum*, é determinado como a soma do valor esperado da probabilidade de seleção dos indivíduos da população atual ou como a soma dos graus de aptidão desses. Os indivíduos são distribuídos no intervalo [0,*Sum*] proporcionalmente ao valor esperado de cada um ou, no segundo caso, ao seu grau de aptidão. Por exemplo, na figura A.2, a circunferência da roleta corresponde à soma dos graus de aptidão dos seis indivíduos da população, representados pelos seus respectivos arcos.

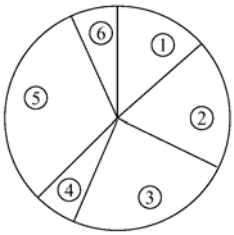


Figura A.2 – Exemplo de Roleta

Nota-se que o indivíduo ‘5’ possui maior grau de aptidão, ocupando consequentemente o maior intervalo, enquanto os indivíduos ‘4’ e ‘6’, com menor aptidão, ocupam intervalos menores da roleta.

Para selecionar um indivíduo, um número aleatório é gerado no intervalo $[0, Sum]$, o que corresponde a um giro da roleta até esse número. O indivíduo correspondente ao intervalo onde a roleta pára é o selecionado. Esse processo é repetido até que se atinja o número de indivíduos desejados para compor a nova população.

O método básico de seleção utilizando roleta é o **SSR** (*Stochastic Sampling with Replacement*) cujos tamanhos dos intervalos e probabilidade de seleção permanecem inalterados durante todo o processo de seleção. O SSR apresenta *bias* igual a zero porém um *spread* potencialmente ilimitado, pois qualquer indivíduo com intervalo não nulo tem a possibilidade de formar, por completo, a próxima população.

O método **SSPR** (*Stochastic Sampling with Partial Replacement*) é uma versão melhorada do SSR, uma vez que o intervalo da roleta correspondente a um indivíduo é reduzido por um fator conforme o mesmo é selecionado. Se o tamanho do segmento

tornar-se negativo, o mesmo é fixado em zero. Isso provê um limite superior ao *spread*, $\lceil et(i) \rceil$. No entanto, o limite inferior $\lfloor et(i) \rfloor$ é zero e o *bias* é maior do que no método SSR.

Os métodos de seleção SSR e SSPR que utilizam roleta podem, geralmente, ser implementados com uma complexidade de tempo da ordem de $N \log N$, onde N é o tamanho da população.

Método SUS (*Stochastic Universal Sampling*)

O método SUS, proposto por Baker [27], consiste na amostragem dos elementos da população utilizando-se um único giro de uma roleta, com características de *spread* mínimo e *bias* zero. A roleta é construída tomando-se por base os graus de aptidão dos elementos da população, que definem, proporcionalmente, o tamanho de cada uma das seções conforme figura A.3 abaixo:

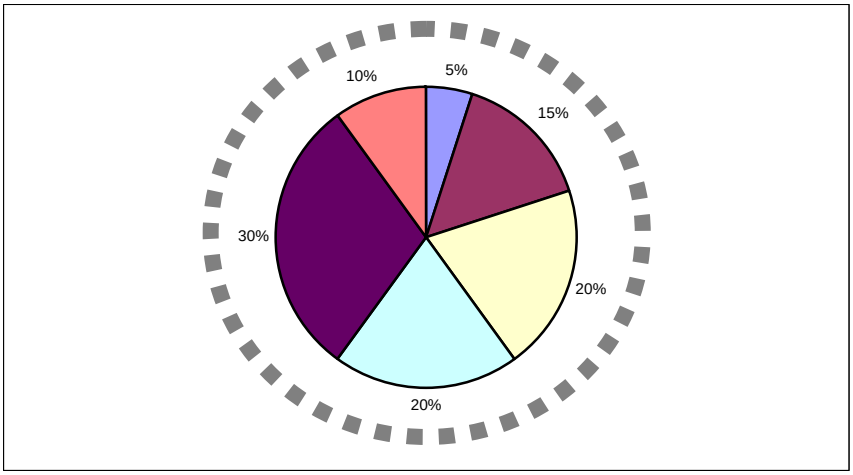


Figura A.3 – Roleta utilizada no Método de Seleção (SUS)

Ao invés do uso de um único ponteiro de seleção utilizado nos métodos da roleta SSR e SSPR, o SUS faz uso de N ponteiros igualmente espaçados ao redor da roleta, em que N é o número de indivíduos a serem selecionados. Esse processo é realizado por meio da geração de um número randômico, ptr, pertencente ao intervalo $[0, Sum/N]$. Os N indivíduos são, então, escolhidos gerando-se N ponteiros espaçados de 1,

$\lceil ptr, ptr+1, ..., ptr+N+1 \rceil$ e selecionando-se aqueles cujos intervalos da roleta são interceptados por esses ponteiros.

Nesse método, é possível garantir que um indivíduo será selecionado, no mínimo, $\lfloor et(i) \rfloor$ vezes e não mais do que $\lceil et(i) \rceil$, alcançando-se dessa forma o *spread* mínimo. Além disso, como a amostragem dos indivíduos é realizada ao redor de toda a população, pode-se afirmar que o método SUS possui *bias* igual a zero.

Em termos de eficiência, a complexidade de tempo do método SUS é da ordem de *N*, o que o torna mais simples e ágil que os métodos SSR e SSPR cuja ordem é de *N*log*N*.