UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO E SISTEMAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

UM MÉTODO DE SOLUÇÃO HEURÍSTICO PARA A PROGRAMAÇÃO DE EDIFÍCIOS DOTADOS DE MÚLTIPLOS PAVIMENTOS-TIPO

Tese Submetida à Universidade Federal de Santa Catarina Para a Obtenção do Título de Doutor em Engenharia de Produção

Jorge de Araújo Ichihara

Florianópolis
1998

UM MÉTODO DE SOLUÇÃO HEURÍSTICO PARA A PROGRAMAÇÃO DE EDIFÍCIOS DOTADOS DE MÚLTIPLOS PAVIMENTOS-TIPO

Jorge de Araújo Ichihara

Esta Tese Foi Julgada Adequada Para a Obtenção do Título de

DOUTOR EM ENGENHARIA DE PRODUÇÃO

e Aprovada em Sua Forma Final pelo Programa de Pós-Graduação em Engenharia de Produção.

Prof. Ricardo Miranda Barcia
Coordenador do Curso

BANCA EXAMINADORA:

Prof. Fernando Álvaro Ostuni Gauthier, Dr.
Orientador

Prof. Carlos Alberto Pereira Soares, Dr.
Examinador Externo

Prof. José Rodrigues de Farias Filho, Dr.
Examinador Externo

Profª. Édis Mafra Lapolli, Drª.

Prof. Oscar Civo López Vaca, Dr.

Prof. Ariovaldo Bolzan, Dr.
Moderador

Dedicatória:

Karin Gonçalves Ichihara

Matthaeus Alexius de Gonçalves Ichihara

Shoji Ichihara

Terezinha de Jesús Queiroz de Araújo Ichihara

Homenagens Póstumas:

Prof. Carl Vicente Limmer
CPGEC / UFF

Prof. Plínio Stange
EPS / UFSC

AGRADECIMENTOS

Ao Orientador:

Fernando Álvaro Ostuni Gauthier

Aos Professores:

Antônio Edésio Jungles
Carlos Alberto Pereira Soares
Édis Mafra Lapolli
José Rodrigues de Farias Filho
Luis Fernando Malmann Heineck
Oscar Ciro López Vaca
Paulo Maurício Selig

Aos Amigos:

Antônio Valdivia Altamirano
Eva Cardoso Gonçalves
Ivandi Silva Teixeira
João Medeiros Tavares Junior
José Ângelo Nicácio
José Hélio Alvarez Elarrat
Maria das Graças de Araújo Cardoso
Regina Cleide Teixeira
Ricardo Mendes Junior
Samuel Cerejo Gonçalves
Selma Maria Amorim Tavares
Silmara Maria Agostini Mendes

Financiamento:

CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

SUMÁRIO

CAPÍTULO I - INTRODUÇÃO

1.1.	Origem do Trabalho	001
1.2.	Objetivos do Trabalho	004
1.3.	Limitações do Trabalho	004
1.4.	Justificativa do Trabalho	005
1.5.	Estrutura do Trabalho	006

CAPÍTULO II – PROGRAMAÇÃO DE PROJETOS

2.1.	Introdução	007
2.2.	Conceito de Projeto	007
2.3.	Definição de Grafo Dirigido	007
2.4.	Conceito de Programa e Programação	008
2.5.	O Gráfico de Gantt	009
2.6.	Técnicas de Programação com Redes	010
2.6.1.	As Redes CPM, PERT e PDM	010
2.7.	Os Problemas de Alocação de Recursos	012
2.7.1.	O Problema do Nivelamento de Recursos	012
2.7.2.	O Problema da Compressão de Projetos	013
2.7.3.	O Problema de Programação de Projetos com Limitação de Recursos	014
2.8.	Caracterização Geral da Programação de Projetos com limitação de Recursos	015
2.9.	A Literatura da Programação de Projetos com Limitação de Recursos	017
2.10.	Procedimentos para Solucionar a Programação de Projetos com Limitação de Recursos	019
2.10.1.	Heurísticas de Programação Baseadas em Regras de Prioridade	021
2.10.1.1.	Critérios de Priorização de Atividades	025
2.10.1.2.	Critérios de Priorização de Modos	025
2.10.2.	Busca Local (*Local Search* ou *Neighbourhood Search*)	025
2.10.3.	Programação Evolucionária (*Evolutionary Programming*)	026
2.11.	Os Projetos Lineares de Construção Civil	027
2.12.	O Uso das Técnicas de Rede na Programação de Projetos Lineares	027
2.13.	Métodos de Sequenciamento de Trabalho	028
2.14.	Características dos Agrupamentos de Fluxos	029
2.15.	O Desenvolvimento dos Métodos de Programação de Projetos Lineares	030
2.16.	O Método da Linha de Balanço (*LOB – Line of Balance Method*)	031
2.17.	A Curva de Agregação de Recursos	032
2.18.	Conclusão	033

CAPÍTULO III – ALGORITMOS GENÉTICOS

3.1. Introdução 035
3.2. A Literatura dos Algoritmos Genéticos 035
3.3. Quando é Interessante Utilizar um Algoritmo Genético 037
3.4. Vantagens dos Algoritmos Genéticos 037
3.5. A Robustez dos Algoritmos Genéticos 038
3.6. Terminologia dos Algoritmos Genéticos 039
3.7. Representação da Estrutura de Solução 040
3.8. Esquema de Cromossomas 041
3.9. Avaliação das Soluções Candidatas 041
3.10. Operadores Genéticos 043
3.10.1. Seleção 044
3.10.1.1. Mecanismos de Seleção 044
3.10.1.2. O Processo de Substituição dos Indivíduos 046
3.10.2. Cruzamento 047
3.10.3. O Operador de Mutação 052
3.10.4. O Operador de Inversão 054
3.11. Planos de Reprodução 054
3.12. Decisões para Implementar um Algoritmo Genético 056
3.13. A Estrutura Básica do Algoritmo 057
3.14. Conclusão 057

CAPÍTULO IV – O MÉTODO PROPOSTO

4.1. Introdução 059
4.2. Definição do Problema 059
4.3. O Método de Solução Proposto 061
4.4. O Modelamento do Projeto 062
4.4.1. Validação do Grafo Atividade-No-Nó 064
4.4.2. Tipo de Ligação Previsto entre Atividades 065
4.4.3. A Numeração dos Pavimentos 065
4.4.4. O Método de Programação de Projetos Lineares Adotado 065
4.4.5. Tipo de Programação Adotado para o Método da Linha de Balanço 066
4.4.6. Combinações Possíveis das Atividades Repetitivas em Relação ao Sentido de Execução 067
4.4.7. Quantidade de Trabalho por Pavimento 067
4.4.8. Número de Equipes por Pavimento Repetitivo 068
4.4.9. O Modo de Produção das Atividades 069
4.4.10. Expressões Gráficas Possíveis da Programação pelo Método Proposto. 073
4.5. A Estrutura Conceitual do Método Proposto 073
4.5.1. O Módulo de Programação 074
4.5.1.1. A Priorização das Atividades 075
4.5.1.2. Os Tipos de Recursos Diretamente Restringidos 076
4.5.1.3. A Restrição de Espaço 076
4.5.1.4. A Restrição de Lógica Horizontal 077
4.5.1.5. A Restrição de *Buffer* 082

4.5.1.6. A Restrição de Data Condicionante 083
4.5.1.7. A Restrição de Lógica Vertical 083
4.5.1.8. A Restrição de Recurso Monetário 084
4.5.2. Módulo de Busca 086
4.5.2.1. Hibridização do Algoritmo Genético 086
4.5.2.2. Classificação das Restrições do Problema 087
4.5.2.3. A Utilização dos Algoritmos Genéticos em Problemas com Restrições 087
4.5.2.4. A Função Objetivo 088
4.5.2.5. A Estruturação do Cromossoma 089
4.5.2.6. Níveis de Especialização do Cromossoma 090
4.5.2.7. Geração da População inicial 090
4.5.2.8. Métodos de Seleção Adotados 091
4.5.2.9. Elitismo 094
4.5.2.10. Cruzamento e Inversão 094
4.5.2.11. Mutação 097
4.5.2.12. Tipos de Redes *versus* Tipos de Operadores 098
4.5.2.13. A Verificação do Tipo de Rede 100
4.5.2.14. Estrutura Básica do Algoritmo Genético 102
4.5. Conclusão 103

CAPÍTULO V – EXPERIMENTOS E RESULTADOS

5.1. Introdução 104
5.2. Os Atributos de um Problema de Programação de Projetos 104
5.3. A Definição dos Experimentos 104
5.4. Características Básicas dos Problemas-Teste 106
5.5. As Medidas de Complexidade dos Problemas-Teste 106
5.6. Valores Limites das Características dos Problemas-Teste 108
5.7. Características da Implementação Computacional do Protótipo 108
5.8. O Ajuste dos Parâmetros dos Algoritmos Genéticos 109
5.9. Resultados dos Testes realizados 109
5.10. Gráficos dos Experimentos Realizados 110
5.11. Performance do Algoritmo Genético 113
5.12. Exemplos de Programas Gerados 118

CAPÍTULO VI – CONCLUSÕES E RECOMENDAÇÕES

6.1. Conclusões 133
6.2. Recomendações 136

BIBLIOGRAFIA 137

ANEXO 157

LISTA DE FIGURAS

CAPÍTULO I – INTRODUÇÃO

1.1. Sistematização Funcional Geral dos Edifícios Dotados de Múltiplos Pavimentos-tipo 002

CAPÍTULO II – PROGRAMAÇÃO DE PROJETOS

2.1. Exemplo de Grafo Orientado 008
2.2. O Gráfico de Gantt 010
2.3. Exemplos de Redes CPM ou PERT 011
2.4. Exemplo de Rede PDM 012
2.5. O Problema da Programação de Projetos com Restrição de Recursos 015
2.6. Estrutura Básica em Pseudocódigo da Heurística de Programação Baseada em Regra de Prioridade 021
2.7. Esquema de Programação Serial em Pseudocódigo 022
2.8. O Algoritmo de Brooks como Exemplo do Esquema de Programação Paralela 024
2.9. Estrutura Básica em Pseudocódigo do Método de Busca de Vizinhança Genérico 026
2.10. Estrutura Básica dos Algoritmos Evolucionários 027
2.11. Representação de um Projeto Repetitivo na Rede CPM 028
2.12. Tipos de Sequenciamento de Trabalho 029
2.13. Métodos de Linha de Fluxo 029
2.14. O Método da Linha de Balanço 032
2.15. Exemplo de Histograma de Recursos, Curva de Agregação de Recursos e Curva de Agregação de Recursos Acumulados ou Curva *S*. 032
2.16. Esboço de Alguns Padrões Teóricos da Curva *S* 033

CAPÍTULO III – ALGORITMOS GENÉTICOS

3.1. Estrutura de um Cromossoma em Codificação Binária 041
3.2. Estrutura Básica de um Algoritmo Genético em Pseudocódigo 057

CAPÍTULO IV – O MÉTODO PROPOSTO

4.1. Tipos de Lógica em um Modelo de Grafos 063
4.2. Integração entre o Grafo Representativo do Projeto e o Método de Programação Linear representativo das Atividades Repetitivas 064
4.3. Erros em um Grafo Atividade-No-Nó 065
4.4. Programação de Atividades Repetitivas 066
4.5. Tipos de Programação pelo Método da Linha de Balanço 067
4.6. Combinações Possíveis entre os dois Sentidos das Atividades

	Repetitivas, segundo a Direção Vertical	067
4.7.	Exemplo de Atividades Repetitivas de Projeto Repetitivo Típico e de Projeto Repetitivo Não Típico	068
4.8.	Exemplo de Atividades com Número Constante de Equipes por Pavimento e Variação do Número de Equipes nos Pavimentos	068
4.9.	Esquematização do Modo de Produção de uma Atividade	069
4.10.	Ilustração das Durações das Atividades de Acordo com o Modo de Produção	071
4.11.	Esboço de uma Possível Expressão Gráfica Utilizando Dados Fornecidos pelo Método Proposto	073
4.12.	Componentes Gerais do Método Proposto	074
4.1.3.	Esquema de Programação Serial em Pseudocódigo	075
4.14.	Exemplo de Priorizações Viáveis e Inviáveis em um Esquema de Programação Serial	076
4.15.	Ligações entre Atividades Não Repetitivas: Exemplificação do Caso 1	077
4.16.	Ligações entre Atividades de Naturezas Diferentes: Exemplificação do Caso 2	078
4.17.	Ligações entre Atividades de Naturezas Diferentes: Exemplificação do Caso 3	078
4.18.	Restrição Horizontal: Exemplificação do Caso 4	079
4.19.	Restrição Horizontal: Exemplificação do Caso 5	080
4.20.	Restrição Horizontal: Exemplificação do Caso 6	080
4.21.	Restrição Horizontal: Exemplificação do Caso 7	081
4.22.	Restrição Horizontal: Exemplificação do Caso 8	081
4.23.	Restrição Horizontal: Exemplificação do Caso 9	082
4.24.	Restrição de *Buffer*	082
4.25.	Exemplificação de Restrição de Lógica Vertical	084
4.26.	Lógica de Restrição Vertical	084
4.27.	Restrição de Recurso Monetário em Pseudocódigo	085
4.28.	Incremento das Despesas da Atividade Atual no Vetor de Despesas Mensais do Projeto	086
4.29.	Representação do Cromossoma	089
4.30.	Níveis de Especialização do Cromossoma no Método Proposto	090
4.31.	Geração da População Inicial	091
4.32.	Seleção pelo Giro da Roleta	093
4.33.	Pseudo-Algoritmo Utilizado para a Reprodução com Elitismo	094
4.34.	Grafo Atividade-No-Nó para um Projeto de Oito Atividades	095
4.35.	Esquema de Operações Genéticas para Redes Mistas	099
4.36.	Esquema de Operações Genéticas para Redes Seriadas	100
4.37.	Exemplo do Conceito de Paralelismo entre os Nós de um Grafo	102
4.38.	Estrutura Básica do Algoritmo Genético em Pseudocódigo	102

CAPÍTULO V – EXPERIMENTOS E RESULTADOS

5.1.	Gráfico Número de Atividades x Tempo de Processamento Computacional: Problemas-teste 1 a 4	111
5.2.	Gráfico Número de Atividades x Função	

	Objetivo: Problemas-teste 1 a 4	111
5.3.	Gráfico Complexidade CNC x Tempo de Processamento Computacional: Problemas-teste 1 a 4	111
5.4.	Gráfico Complexidade CNC x Função Objetivo: Problemas-teste 1 a 4	112
5.5.	Gráfico Complexidade T-density Médio x Tempo de Processamento Computacional: Problemas-teste 1 a 4	112
5.6.	Gráfico Complexidade T-density Médio x Função Objetivo: Problemas-teste 1 a 4	112
5.7.	Evolução do Melhor Valor da Função Objetivo no Problema 1	113
5.8.	Evolução do Valor Médio da Função Objetivo no Problema 1	113
5.9.	Evolução do Valor da Função Objetivo no Problema 2	114
5.10.	Evolução do Valor Médio da Função Objetivo no Problema 2	114
5.11.	Evolução do Melhor Valor da Função Objetivo no Problema 3	114
5.12.	Evolução do Valor Médio da Função Objetivo no Problema 3	115
5.13.	Evolução do Melhor Valor da Função Objetivo no Problema 4	115
5.14.	Evolução do Valor Médio da Função Objetivo no Problema 4	115
5.15.	Evolução do Melhor Valor da Função Objetivo no Problema 5	116
5.16.	Evolução do Valor Médio da Função Objetivo no Problema 5	116
5.17.	Evolução do Melhor Valor da Função Objetivo no Problema 6	116
5.18.	Evolução do Valor Médio da Função Objetivo no Problema 6	117
5.19.	Evolução do Melhor Valor da Função Objetivo no Problema 7	117
5.20.	Evolução do Valor Médio da Função Objetivo no Problema 7	117
5.21.	Evolução do Melhor Valor da Função Objetivo no Problema 8	118
5.22.	Evolução do Valor Médio da Função Objetivo no Problema 8	118
5.23.	Histograma dos Recursos Monetários Disponíveis para o Problema 3	119
5.24.	Perfil dos Recursos Monetários para o Melhor Experimento do Problema 3	121
5.25.	Gráfico dos Desvios entre os Recursos Monetários Mensais Disponíveis e Requeridos para o Melhor Experimento com o Problema 3	121
5.26.	Perfil dos Recursos Monetários para o 2° Melhor Experimento Com o Problema 3	123
5.27.	Gráfico dos Desvios entre os Recursos Monetários Mensais Disponíveis e Requeridos para o 2° Melhor Experimento com o Problema 3	123
5.28.	Perfil dos Recursos Monetários para o 3° Melhor Experimento Com o Problema 3	125
5.29.	Gráfico dos Desvios entre os Recursos Monetários Mensais Disponíveis e Requeridos para o 3° Melhor Experimento com o Problema 3	125
5.30.	Perfil dos Recursos Monetários para o Melhor Experimento Com o Problema 5	127
5.31.	Gráfico dos Desvios entre os Recursos Monetários Mensais Disponíveis e Requeridos para o Melhor Experimento com o Problema 5	127
5.32.	Perfil dos Recursos Monetários para o 2° Melhor Experimento Com o Problema 5	129
5.33.	Gráfico dos Desvios entre os Recursos Monetários Mensais	

Disponíveis e Requeridos para o 2° Melhor Experimento com o Problema 5 129
5.34. Perfil dos Recursos Monetários para o 3° Melhor Experimento Com o Problema 5 131
5.35. Gráfico dos Desvios entre os Recursos Monetários Mensais Disponíveis e Requeridos para o 3° Melhor Experimento com o Problema 5 131

LISTA DE TABELAS

CAPÍTULO IV – O MÉTODO PROPOSTO

4.1. Tabela dos Tipos de Rede *Versus* Operadores do Modelo Proposto..... 100

CAPÍTULO V – EXPERIMENTOS E RESULTADOS

5.1. Características Básicas dos Problemas-teste..... 106
5.2. Medição da Complexidade dos Problemas-teste..... 108
5.3. Resumo das Características dos Problemas-teste em seus Valores Limites..... 108
5.4. Resultado dos Experimentos Realizados usando os Problemas-teste..... 110
5.5. Recursos Monetários Disponíveis para o Problema 3..... 119
5.6. Programa que Conduz ao Resultado Considerado Ótimo para o Problema 3..... 120
5.7. Melhor Resultado de Recursos Monetários para o Problema 3..... 121
5.8. Melhor Programa Referente ao Problema 3..... 122
5.9. 2° Melhor Resultado de Recursos Monetários para o Problema 3..... 123
5.10. 2° Melhor Programa Referente ao Problema 3..... 124
5.11. 3° Melhor Resultado de Recursos Monetários para o Problema 3..... 125
5.12. 3° Melhor Programa Referente ao Problema 3..... 126
5.13. Melhor Resultado de Recursos Monetários para o Problema 5..... 127
5.14. Melhor Programa Referente ao Problema 5..... 128
5.15. 2° Melhor Resultado de Recursos Monetários para o Problema 5..... 129
5.16. 2° Melhor Programa Referente ao Problema 5..... 130
5.17. 3° Melhor Resultado de Recursos Monetários para o Problema 5..... 131
5.18. 3° Melhor Programa Referente ao Problema 5..... 132

RESUMO

Os edifícios dotados de múltiplos pavimentos-tipo são caracterizados por muitas unidades repetitivas ou pavimentos típicos. Este tipo de construção possui algumas vantagens decorrentes da repetição do trabalho, tais como a melhoria da produtividade e a redução dos tempos de espera. Consequentemente, muitas das filosofias e metodologias de programação de projetos originadas na Engenharia Industrial tem sido utilizadas neste campo de pesquisa.

O presente trabalho apresenta um método de solução para o Problema da Programação de Projetos com Restrição de Recursos, adaptado para projetos de construção deste tipo de edifício. As atividades inerentes são modeladas por um grafo dirigido e pelo Método da Linha de Balanço, podendo ser repetitivas ou não repetitivas. Os objetivos de programação consistem em minimizar os desvios mensais entre os recursos monetários disponíveis e as despesas, bem como promover a continuidade do trabalho nas atividades repetitivas.

O método proposto utiliza como processo de alocação de recursos o Esquema de Programação Serial, o qual consiste em uma heurística baseada em regras de prioridade; e utiliza como método de busca os Algoritmos Genéticos, os quais possuem a capacidade de localizar um ponto ótimo global em uma região multimodal. Esta hibridização permite a melhoria progressiva de um conjunto de soluções iniciais, através da manipulação do sequenciamento e dos múltiplos modos de produção das atividades.

Os testes realizados através de um protótipo computacional desenvolvido, demonstram que o algoritmo é viável em termos de eficiência e efetividade. Nesta etapa da pesquisa, foram utilizados 8 problemas-teste, constituídos por projetos de construção reais, em séries de 30 experimentos, atingindo um total de 240 experimentos.

ABSTRACT

Multistory buildings are characterized by many repetitive units or typical floors. This type of construction possesses some advantages resulting from the repetitive nature of the work, such as the improvement of the productivity and the reduction of the set up times. Consequently, many of the projects programming philosophies and methodologies which result from Industrial Engineering have been used in this field of research.

This current work presents a solution method for the Resource Constrained Project Scheduling Problem, adapted for construction projects of this building type. The inherent activities are modeled by a directed graph and by the Line of Balance Method, and these activities may be repetitive or non-repetitive. The programming objectives are to minimize the monthly deviations between available monetary resources and the expenses, as well as to promote the continuity of labor in repetitive activities.

The proposed method uses the Serial Schedule Scheme as the resource allocation process, which consists of a priority rule-based scheduling heuristic; and it uses the Genetic Algorithms as a search method, which possess the capacity to locate the global optimum in a multimodal landscape. This hybridization allows the progressive improvement of a group of initial solutions, through the manipulation of the sequencing and of the several activities production modes.

The tests conducted through a developed computerized prototype demonstrate that the algorithm is viable in terms of efficiency and effectiveness. In this stage of research, 8 problem-tests were used, consisting of real construction projects, in a series of 30 experiments, reaching a total of 240 experiments.

CAPÍTULO I

INTRODUÇÃO

Este primeiro capítulo possui a finalidade de apresentar em linhas gerais o presente trabalho de tese. Assim, inicialmente relata-se a origem da pesquisa, destacando-se a definição do problema abordado, em forma de pergunta; a seguir encontram-se a síntese dos objetivos e as limitações, bem como pode-se observar o item referente à justificativa. Finalmente, mostra-se a seqüência dos capítulos que compõem esta pesquisa.

1.1. Origem do Trabalho

Um dos empreendimentos de Construção Civil que mais prolifera nos centros urbanos - a produção de edifícios dotados de múltiplos pavimentos repetitivos - vem merecendo especial atenção de empreendedores e pesquisadores nos últimos anos, devido ao seu grande potencial de racionalização. Similarmente às obras de pontes, oleodutos ou estradas, pode-se tirar partido da repetitividade de muitas partes importantes, em analogia com a produção em massa da indústria fabril; neste sentido em particular, a produção da torre do edifício assemelha-se a uma linha de produção, e daí pode-se adaptar e utilizar uma gama de conhecimentos altamente refletidos e experimentados.

É no campo da Administração de Projetos, mais precisamente no processo de programação (*scheduling*) da produção, que esta pesquisa explora os benefícios extraídos da repetitividade do trabalho; mas, os problemas a serem resolvidos constituem desafios muitas vezes intransponíveis à formulação analítico-matemática, devido à sua inerente natureza complexa e combinatorial. Nos problemas do mundo real existem múltiplos objetivos (às vezes contraditórios), bem como um grande número de variáveis (qualitativas e quantitativas) e condições de contorno, que explodem em um campo de risco e incerteza. Portanto, é impossível modelar um problema em toda a sua plenitude e tampouco criar um método de solução que consiga resolvê-lo, daí surgirem inúmeras propostas dotadas de hipóteses e simplificações, que consideram somente as partes do problema julgadas mais relevantes e suficientes para auxiliar na tomada de decisão.

Um edifício de múltiplos pavimentos-tipo pode ser dividido em cinco sistemas físicos (fig. 1.1): as fundações, o subsolo (garagens, maquinarias, central de instalações, *etc.*), o térreo, os pavimentos não típicos (garagens, *play-grounds*, salões para eventos sociais, *etc.*), os pavimentos típicos (escritórios ou unidades residenciais), e as obras de cobertura (caixa d'água, cobertura, elevador, *etc.*); acrescenta-se mais as obras externas (cisternas, piscinas, ajardinamento, *etc.*) para compor um sexto sistema.

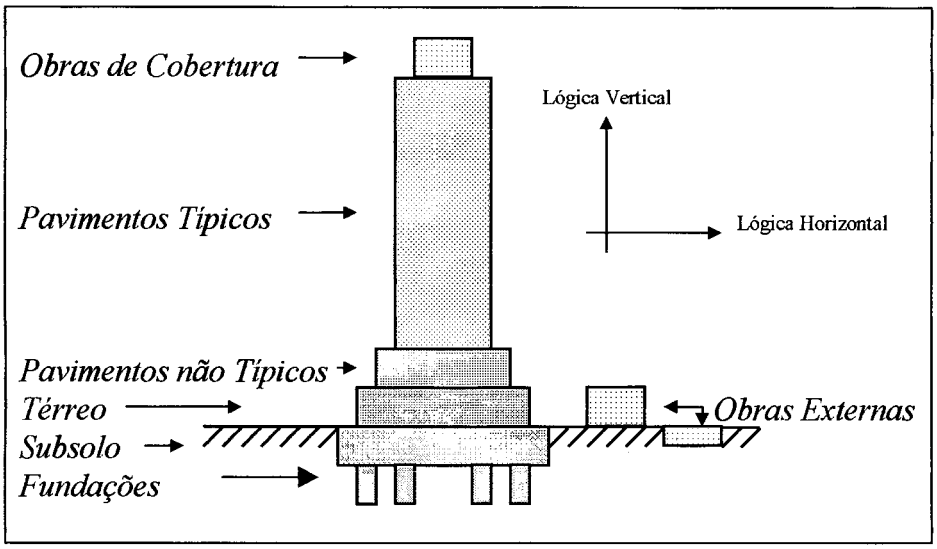


Fig. 1.1 - Sistematização Funcional Geral dos Edifícios Dotados de Pavimentos-tipo.
Fonte: do Autor.

Existem dois tipos de atividades a serem programadas: (1) àquelas que repetem-se no sentido vertical da construção, e que são denominadas *repetitivas* (estrutura, alvenaria e pintura da torre, entre outras); e (2) àquelas que são realizadas em apenas um dos pavimentos da construção, e por este motivo são denominadas *não repetitivas* (fundações, subsolo, térreo, pavimentos não típicos, obras de cobertura e obras externas). Ambas as atividades podem estar dispostas em série ou em paralelo e suas durações podem variar segundo o incremento ou decréscimo do número de equipes de trabalho. Uma característica fundamental para consideração é que existem muitos benefícios, inclusive de motivação, produtividade e balanceamento de recursos, quando as atividades repetitivas são executadas sem interrupção (mudança temporária para outra atividade).

De acordo com a literatura, este problema de programação envolve seis tipos de restrições diferentes: (1) de datas condicionantes para iniciar uma atividade; (2) de precedências entre atividades no mesmo piso, tradicionalmente conhecidas através das técnicas de rede CPM (*Critical Path Method*) e PDM (*Precedence Diagram Method*); (3) de *buffers* (intervalos de tempo) impostos entre duas atividades; (4) de precedências entre atividades em pisos diferentes, que é uma lógica própria das técnicas de programação de atividades repetitivas; (5) de limitação da quantidade de equipes destinadas a uma atividade, de acordo com o espaço disponível; e (6) de recursos monetários previstos por período de tempo.

Os empreendimentos cujos fundos destinados à construção, são decorrentes principalmente das vendas das unidades residenciais ou comerciais, em geral possuem um horizonte maior de produção, que é função da velocidade das vendas; a tendência geral de programação, é que as despesas sejam adiadas tanto quanto possível. Os empreendimentos cujos fundos decorrem mais significativamente de financiamento, possuem um horizonte menor de produção, já que a conclusão no menor prazo é interessante ao empreendedor, na intenção de pleitear novos financiamentos, fazer girar

o capital mais rapidamente, bem como para antecipar o repasse do financiamento aos adquirentes das unidades e antecipar o recebimento de sua parte na poupança; a tendência geral é que as despesas sejam niveladas aos máximos valores periódicos possíveis. Por sua vez, os empreendimentos empreitados estão condicionados à capacidade de pagamento do contratante, e seguem a mesma tendência de programação do empreendimento financiado, de modo que os pagamentos pelo progresso da obra sejam sempre altos.

Em conseqüência, a programação das atividades que compõem o projeto, pode ser feita segundo um plano de distribuição das despesas diretas ao longo do tempo [Heineck, 1988]. A Curva de Agregação de Recursos, consiste em uma ferramenta típica para esta finalidade: de acordo com dados estatísticos e/ou modelamento matemático, a empresa pode traçar um perfil para as despesas diretas do projeto, que seja compatível com o tipo de captação dos recursos monetários, bem como com os planos de produção aos níveis estratégico, tático e operacional. Neste sentido, são levadas em conta variáveis como a previsão dos fluxos de caixa da empresa (para a formação do capital de giro), os vários projetos em andamento e os aspectos conjunturais. Neste contexto, surge o conceito de curvas "S" padronizadas, como a curva à 50%, em que a metade das despesas devem ser realizadas até a metade da duração total do projeto; em tese, existe uma família de curvas, das quais pelo menos uma representa perfeitamente os interesses da empresa.

Considerando a existência de uma curva de agregação de recursos monetários para o projeto (a qual não será discutida ou pesquisada neste trabalho), ou de outra ferramenta semelhante, o trabalho de programação das atividades consiste em gerar um contexto das atividades cujos montantes de despesas diretas a cada período de tempo, possuam a menor diferença possível em relação aos equivalentes na curva planejada.

Finalmente, considerando todas as características expostas e situando-se no campo determinístico, pode-se definir o problema a ser abordado neste trabalho através da seguinte pergunta: -**"De posse de uma Curva de Agregação de Recursos Monetários acumulados por período de tempo, previamente estimada para a construção de um edifício dotado de múltiplos pavimentos-tipo, como desenvolver um método de solução que seja capaz de gerar um programa o qual apresente um perfil de distribuição de despesas diretas o mais próximo possível da curva planejada, mantendo a continuidade das equipes de trabalho nas atividades repetitivas?**

É inviável o uso de métodos analíticos ou exatos para solucionar este tipo de problema. Uma solução possível pode ser a utilização de uma classe heurística muito pesquisada nos últimos anos, as denominadas meta-heurísticas, como os Algoritmos Genéticos e o *Simulated Annealing*, as quais possuem características e propriedades que lhes confere a habilidade de serem mais *insensíveis* aos pontos mínimos locais que a maioria das heurísticas tradicionais, além de adequarem-se perfeitamente aos problemas combinatoriais. A escolha dos Algoritmos Genéticos como heurística de busca local para este trabalho, deve-se principalmente a uma de suas características mais importantes: - a propriedade do paralelismo implícito - a qual possibilita analisar, avaliar e evoluir um

conjunto de soluções simultaneamente; isto é, estes algoritmos trabalham com uma superfície, e não somente com uma região desta superfície.

1.2. Objetivos do Trabalho

Este trabalho de pesquisa possui o seguinte objetivo principal:

- Apresentar contribuição à pesquisa de programação de projetos lineares, ao apresentar um método capaz de: (1) programar segundo uma filosofia orientada aos recursos monetário e de mão-de-obra; (2) permitir interligações de atividades em série e em paralelo; (3) integrar atividades repetitivas e não repetitivas em um mesmo procedimento; (4) pesquisar múltiplos modos de execução para as atividades, através do incremento de equipes de trabalho; (5) admitir atividades repetitivas com dois sentidos de execução, o ascendente e o descendente; (6) programar as atividades repetitivas segundo ritmos naturais de trabalho; (7) considerar os seis tipos de restrições julgados principais; e (8) programar eficientemente projetos de tamanho grande.

Outros objetivos consistem em:

- Apresentar um método de solução fundamentado em heurísticas capazes de escapar dos pontos mínimos locais;
- Introduzir um método de solução que seja interessante à futuros desenvolvimentos de pacotes de *softwares* comerciais, devido a sua efetividade, eficiência e a possibilidade de resultar em interfaces amigáveis com o usuário final.

1.3. Limitações do Trabalho

As limitações são basicamente:

- A abordagem é construída no campo das variáveis quantitativas, e de variáveis qualitativas que podem ser mensuradas por meio de uma escala quantitativa;
- A abordagem é de cunho determinístico; não são incluídas análises probabilísticas;
- O método proposto aplica-se à programação de um único projeto de cada vez; não está prevista a análise simultânea de vários projetos;

- A produtividade de uma equipe de trabalho é considerada constante ao longo do tempo, e ao longo da execução das atividades repetitivas;

- O processo de alocação dos recursos utilizados, não considera a interrupção das atividades, mesmo que temporariamente.

1.4. Justificativa do Trabalho

A Construção Civil é uma atividade econômica baseada em projetos. Além disto, seus empreendimentos possuem algumas características (o ineditismo e a longa duração de cada projeto, o alto custo unitário de cada obra, a grande rotatividade da mão-de-obra, o canteiro de obras fisicamente afastado da empresa, *etc.*) que tornam a programação uma das tarefas potencialmente mais importantes para a sobrevivência nesta indústria. Este fato é ampliado à medida em que o ambiente se torna mais competitivo, face às mudanças decorrentes dos fenômenos da globalização e da abertura dos mercados.

Considerando este contexto, é notório que o setor de planejamento das empresas construtoras necessita de ferramentas apropriadas que auxiliem no processo decisório. Existem duas maneiras puras de tomar uma decisão: a intuitiva, baseada na convicção ou preferência pessoal; e a racional, única sustentável à rigor, fundamentada no raciocínio e no desenvolvimento do cálculo. A pesquisa acerca de modelos e métodos racionais, orientados para a solução de problemas importantes e complexos sob algum ponto de vista, contribui para aumentar a consistência das decisões, e conseqüentemente para a sobrevivência da empresa.

Na literatura técnica dos projetos de natureza repetitiva, não existem métodos de solução analíticos eficientes envolvendo restrições de recursos e múltiplos modos de execução das atividades: o Problema da Programação de Projetos com Restrição de Recursos *(RCPSP: Resource-Constrained Project Scheduling Problem)*, insere-se na classe de problemas combinatoriais NP-completos [Wiest, 1967; Boctor, 1990; Chan *et al.*, 1996; Cho e Kim, 1997]; em conseqüência, os métodos de solução enumerativos, assim como os analíticos, servem apenas aos pequenos projetos teóricos, frente à multiplicidade de restrições e dados referentes ao projeto. Portanto, os meios mais eficientes para obter soluções viáveis são atualmente obtidos através de heurísticas [Davis e Patterson, 1975].

Também não se tem conhecimento de métodos de solução direcionados para a programação de edifícios com múltiplos pavimentos-tipo, os quais incorporem conjuntamente todas as características evidenciadas nos itens anteriores deste capítulo. Estas características tornam a programação mais realística, embora aumentem significativamente o esforço computacional na implementação. A promoção da continuidade do trabalho nas atividades repetitivas, por sua vez, contribui para minimizar os tempos de espera de equipamentos, materiais e mão-de-obra, ao mesmo tempo em que minimiza os problemas relativos às interrupções de trabalho, como as operações de armazenamento temporário de material e ferramentas e os tempos de preparação (*Setup time*), problemas tão comuns e prejudiciais à construção.

Devido a dificuldade encontrada para resolver problemas deste tipo, com uma considerável redução de hipóteses e simplificações, existe um grande interesse no meio científico por métodos novos de solução. Os algoritmos heurísticos como os Algoritmos Genéticos, o *Simulated Annealing*, e o *Tabu Search*, oriundos da Inteligência Artificial, tem ocupado um espaço significativo nos últimos anos, devido à inúmeras qualidades como: amplas possibilidades de contornar os pontos mínimos locais, robustez, eficiência e capacidade para solucionar problemas complexos e de grande porte.

Bjornal *et al.*, (1995), acrescentam que este é um tempo importante para pesquisar otimização combinatorial, considerando que parece provável que novas e mais poderosas meta-heurísticas surgirão nos próximos anos, de forma amplamente disponível aos usuários em pacotes de *softwares* comerciais e bibliotecas. Isto evidencia a necessidade de pesquisá-las e aplicá-las intensamente, de modo a aprofundar o conhecimento sobre elas. Além disso, os mesmos autores acrescentam que a pesquisa acerca destas heurísticas, é importante não somente para o problema em questão, mas também para encontrar soluções aproximadas de um grande número de problemas não acadêmicos, extremamente difíceis, aumentando a integração da academia com a indústria.

1.5. Estrutura do Trabalho

O trabalho está organizado em seis capítulos, sendo este primeiro de introdução ao tema, e os outros seis como descritos a seguir: o segundo capítulo apresenta uma abordagem aos problemas de Programação de Projetos e aos métodos de solução; o terceiro capítulo é dedicado aos Algoritmos Genéticos; o quarto capítulo define o problema e propõe o algoritmo heurístico de solução; no quinto capítulo, encontram-se os experimentos e os resultados obtidos através da aplicação do método em problemas-teste; o sexto capítulo contém as conclusões. No final, encontram-se a bibliografia e o anexo.

CAPÍTULO II

PROGRAMAÇÃO DE PROJETOS

2.1. Introdução

Este capítulo aborda a Programação de Projetos, constituindo-se em um requisito fundamental para o entendimento e o referenciamento bibliográfico do procedimento heurístico de programação de atividades, presente no quarto capítulo deste trabalho de tese.

Sua organização foi elaborada à partir de cinco ênfases implícitas: aos métodos tradicionais de programação de projetos, principalmente as técnicas de rede; aos problemas mais importantes de alocação de recursos, destacando-se o Problema de Programação de Projetos com Restrição de Recursos (RCPSP: *Resource-Constrained Project Scheduling Problem*); aos métodos utilizados na solução do RCPSP, evidenciando-se os procedimentos heurísticos; aos Métodos de Programação Linear, em especial o Método da Linha de Balanço; e finalmente à Curva de Agregação de Recursos.

2.2. Conceito de Projeto

O entendimento à respeito do termo *projeto* varia de acordo com as áreas em que é utilizado. Ferreira (1986) apresenta os seguintes significados no seu dicionário: (1) idéia que se forma de executar ou realizar algo, no futuro; (2) empreendimento a ser realizado dentro de um determinado esquema: *projeto administrativo*, *projeto educacional*; (3) redação ou esboço preparatório ou provisório de um texto: *projeto de estatuto*, *projeto de tese*; (4) esboço ou risco de obra a se realizar; plano: *projeto de cenário*; (5) em Arquitetura, plano geral de edificação.

Os significados (2), (4) e (5) são particularmente interessantes para o presente contexto. Uma possível especialização para os objetivos almejados no campo da Administração de Projetos, pode ser encontrada em Limmer (1991): um "conjunto de estudos e realizações físicas, compreendendo desde a concepção inicial de uma idéia até a sua concretização, traduzida por um empreendimento em operação e pronto para funcionar".

2.3. Definição de Grafo Dirigido

Um grafo $G(p,a)$ consiste de um conjunto de pontos ou nós $p = \{p_1,..., p_n\}$, com $n>0$, e de um conjunto de arcos $a = \{a_1,..., a_m\}$, tal que cada ponto é conectado com pelo menos outro ponto, e cada arco conecta dois pontos de p. O grafo G é dito orientado se todo arco possui um sentido, bem como é denominado finito se m e n são

finitos. Um subconjunto de pontos $\{p_{i1},..., p_{ik}\}$ e arcos $\{a_{j1},...,a_{jk-1}\}$ de um grafo orientado G tal que $a_{j\lambda}$ conecta $p_{j\lambda}$ e $p_{j\lambda+1}$, neste sentido, com $\lambda=\{1,...,k-1\}$, é chamado um caminho de p_{i1} para p_{ik} . Quando $p_{i1}=p_{ik}$, o caminho recebe a denominação de ciclo; em um caso particular, se $k=1$ o caminho é denominado circuito.

- **Definição**: um grafo finito $Gn=(p,a)$, com $n>0$, $p=\{p_1,..., p_n\}$ e $a=\{a_1,..., a_m\}$, é dito dirigido de p_1 para p_n diante de quatro condições: Gn é orientado; a não possui arcos dirigidos para p_1 ou de p_n; Gn não contém ciclos ou circuitos; todo ponto e todo arco estão contidos em pelo menos um caminho dirigido de p_1 para p_n.

A definição implica em que todo arco está contido em um caminho e no mesmo sentido deste, se este caminho vai do nó inicial até o final. Finalmente, um grafo dirigido é denominado completo se todo par de pontos é conectado por pelo menos um arco. Na figura 2.1, mostra-se um grafo orientado composto pelos pontos $\{p_1,..., p_6\}$ e pelos arcos $\{a_1,..., a_9\}$, onde pode-se observar os ciclos $\{a_2, a_3\}$ e $\{a_1,..., a_8,a_9\}$, bem como o circuito $\{a_7\}$. Para torná-lo um grafo dirigido, os arcos a_3, a_7, e a_9 teriam que ser eliminados.

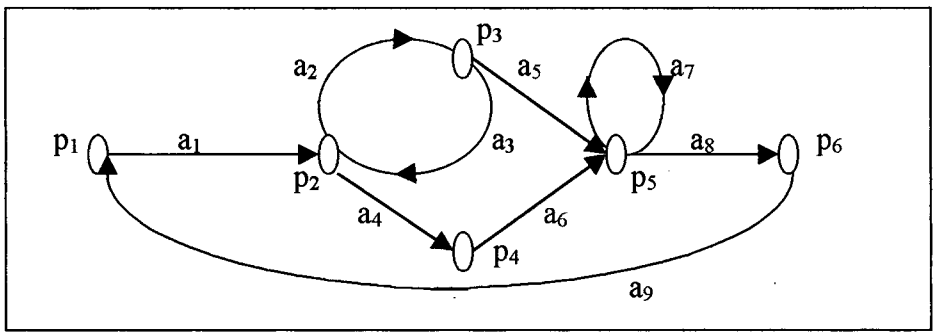


Fig. 2.1 - Exemplo de Grafo Orientado: o conjunto $\{p_1,..., p_6\}$ é constituído pelos pontos ou nós, e $\{a_1,..., a_9\}$ compõe os arcos orientados. Fonte: do autor.

2.4. Conceitos de Programa e Programação

Ferreira (1986), apresenta muitos significados para o termo *programa*, dentre os quais os mais importantes para o contexto abordado são: (1) exposição sumária das intenções ou projetos de um indivíduo, de um partido político, de uma organização, *etc.*, (2) plano, intento, projeto; (3) em Processamento de Dados, a seqüência de etapas que devem ser executadas pelo computador para resolver um problema determinado.

Ao longo desta tese, cada utilização deste termo pode ser enquadrada em uma das duas áreas de conhecimento: Ciência da Computação ou Administração de Projetos. No primeiro caso, o conceito utilizado é o de número (3) [Ferreira, 1986]; no segundo caso, o conceito insere-se no raciocínio de (1) e (2) do mesmo autor, mas fica

oportunamente mais especializado através do termo *schedule* da literatura técnica de língua inglesa, evidenciado nos próximos parágrafos.

Considera-se um projeto complexo que consiste de n atividades, algumas das quais executadas em série e outras em paralelo, tal que este projeto seja representado por um grafo dirigido $G_n(p,a)$. Os nós $p = \{p_1, p_2,..., p_n\}$ representam as atividades, com p_1 e p_n caracterizados como fantasmas (*dummies*); sob tal designação, estas duas atividades possuem apenas finalidade lógica, simbolizando o início e o final do projeto. A_i é o conjunto das atividades precedentes de p_i, com $i = \{1,...,n\}$; analogamente, ξ é o conjunto formado pelos A_i existentes, ou $\xi = \{A_1, A_2,...,A_n\}$, com $A_1 = \phi$. Ambos os conjuntos A_i e ξ, são importantes para estabelecer o seqüenciamento das atividades; isto é, cada atividade só pode iniciar após o término da execução de todas as suas precedentes contidas em A_i. Em conseqüência, ξ governa o andamento do projeto. Neste contexto, os arcos $a = \{a_1, a_2,..., a_m\}$ denotam somente os inter-relacionamentos entre atividades.

- **Definição**: Seja $G_n(p,a)$ um grafo dirigido representando as atividades de um projeto complexo, onde $p = \{p_1,..., p_m\}$ contém as atividades e $a = \{a_1,..., a_m\}$ denota o conjunto das interligações existentes. O programa (*schedule*) $S = (G_n, T_n)$ das atividades consiste do grafo G_n e do conjunto $T = \{T_1,...,T_n\}$ das datas de início das atividades $p_1,...,p_n$, respectivamente [Mayhugh, 1964].

De maneira equivalente, o grafo G_n conjuntamente com o conjunto X_n das durações das atividades $p_1,..., p_n$, respectivamente, também é reconhecido como um programa, representado por $S = (G_n, X_n)$ [Mayhugh, 1964].

A flexão *programação* segundo Ferreira (1986), significa: (1) ato de programar, de estabelecer um programa, (2) o programa ou plano de trabalho de uma empresa, indústria, organização, etc., para ser cumprido em determinado período de tempo; (4) o conjunto dos programas e (4) em processamento de Dados, elaboração de um programa de computador. Todos estes significados são perfeitamente aplicáveis nesta presente pesquisa.

Ashour (1972) evidencia a importância de estabelecer a diferença entre programação (*scheduling*) e seqüenciamento (*sequencing*), dois termos muito confusos na literatura técnica da Administração de Projetos. Segundo o autor, a programação está ligada às especificações das datas de início e de término destas atividades, enquanto o seqüenciamento está ligado aos arranjos e permutações que determinam a ordem de execução das atividades.

2.5. O Gráfico de Gantt

O Gráfico de Gantt foi idealizado pelo engenheiro industrial norte-americano Henry Gantt em 1917 [Harding, 1989], e aplicado na área militar durante a

primeira guerra mundial. Segundo Clark [*apud* Belchior, 1970], a grande contribuição deste gráfico para a arte da decisão, foi a de relacionar os fatos com o tempo.

Pode-se descrevê-lo como sendo um gráfico plano de barras horizontais, onde cada barra representa uma atividade do projeto. Na direção horizontal encontram-se as durações em escala de tempo, e na direção vertical encontram-se as identificações. Como exemplo, na figura 2.2 a seguir, as atividades $\{A_1, A_2, \ldots, A_5\}$ estão programadas obedecendo ao prazo t_5. Quando trata-se de projetos longos e com grande numero de atividades, este tipo de representação torna-se confuso, devido à falta de clareza na visualização dos inter-relacionamentos das atividades e na visão de conjunto.

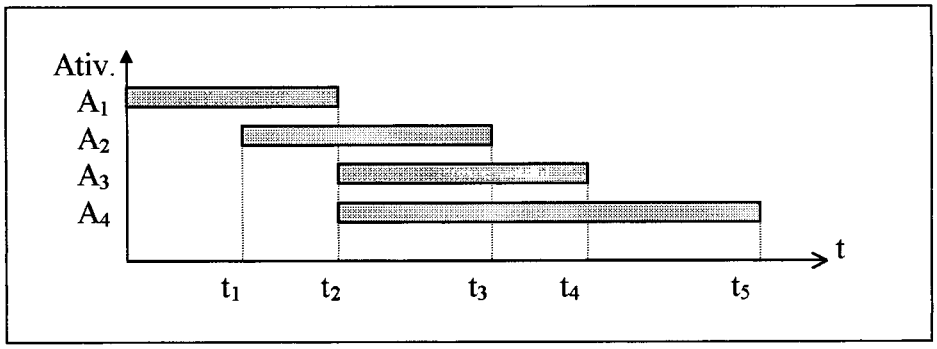


Fig. 2.2 - O Gráfico de Gantt: as barras, que representam as atividades A_i, encontram-se dispostas em escala de tempo t. Fonte: do autor.

2.6. Técnicas de Programação com Redes

A base das técnicas de programação com redes (*network scheduling techniques*) são os diagramas de rede do projeto (*project network diagram*) [Davis, 1973], com seus dois tipos de representação associados a um grafo dirigido: as atividades sendo representadas pelos nós, e alternativamente as atividades sendo representadas pelos arcos. Da mesma forma, os diagramas de redes do projeto são a base da programação orientada pelo tempo, constituídos entre outros, pelos tradicionais métodos CPM, PERT e PDM. Estes métodos são chamados genericamente de Métodos do Caminho Crítico, porque através do cálculo de suas variáveis, é possível diagnosticar o caminho mais longo do grafo, denominado caminho crítico, formado pelas denominadas atividades críticas.

2.6.1. As Redes CPM, PERT E PDM

O CPM (*Critical Path Method*) foi criado em 1957 pela Cia. E. U. du Pont, assessorada pela Remington Rand Division da Sperry Rand Corporation, com o objetivo de aprimorar a programação de projetos ligados às fábricas de produtos químicos. As atividades pertinentes a estes projetos possuíam durações passíveis de serem estimadas com acurácia considerável, fato que resultou na formulação determinística deste método.

No ano seguinte, em 1958, um grupo formado pelo Special Projects Office da marinha norte-americana, pela firma consultora de empresas Booz-Allen & Hamilton International, e pela empreiteira de projéteis balísticos Lockheed, concebeu o PERT (*Critical Path Method*) com o objetivo de programar a fabricação do míssil Polaris. Neste projeto, haviam milhares de peças diferentes, que nunca haviam sido fabricadas em série, o que forçou a introdução do conceito de duração probabilística das atividades.

O PDM (*Precedence Diagram Method*), foi desenvolvido pelo professor Bernard Roy, da Universidade de Sorbonne, no inicio da década de 60. Sua metodologia nasceu da idéia de simplificar o CPM, e posteriormente foi amplamente estudada e divulgada pelo professor John Fondhal, da Universidade de Stanford.

Nas redes CPM e PERT, os nós representam eventos ou acontecimentos e os arcos representam simultaneamente as atividades e os interrelacionamentos. O PDM utiliza os nós como sendo as atividades, e os arcos exercem somente a função de estabelecer os interrelacionamentos. O método de representação do CPM e do PERT é conhecido como Método Americano, e o do PDM, como Método Francês, devido às suas origens. A literatura sobre este métodos é abundante, e pode-se citar Antill e Woodhead (1970), Moder *et al.*, (1983), Willis (1985) e Boiteaux (1985). Na otimização do custo, tem-se os trabalhos pioneiros de Kelley (1961), Fulkerson (1961), e Prager (1963).

Como exemplo, pode-se considerar um projeto P formado pelas seguintes atividades: A (inicial); B, C e D (dependentes de A); E (dependente de D) e F (dependente de B, C e D). Na figura 2.3, mostra-se sua representação segundo as redes CPM ou PERT, onde as atividades $\{A, B, C, D, E, F\}$ recebem as nomenclaturas $\{A_{12}, A_{23}, A_{24}, A_{25}, A_{46}, A_{56}\}$, respectivamente. Uma atividade A_{ij} é caracterizada pelo seu evento inicial i e pelo seu evento final j, contidos no conjunto dos eventos $\{1, 2,..., 6\}$. Este tipo de representação exige a presença de duas atividades fictícias A_{35} e A_{45}, com o objetivo de evitar erros de codificação computacional. À cada A_{ij} corresponde uma duração d_{ij} como atributo principal.

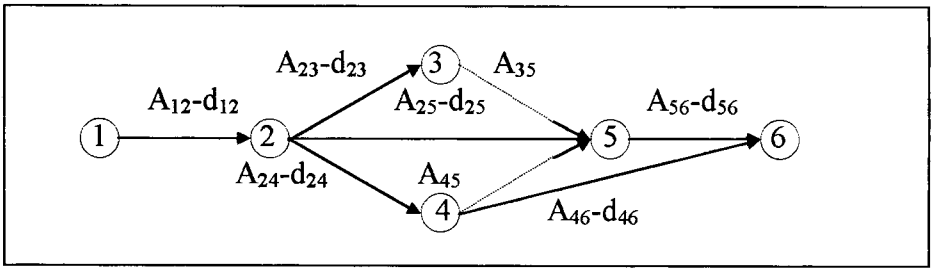


Fig. 2.3 - Exemplo de Redes CPM ou PERT: as atividades A_{ij} são marcadas pelos seus eventos inicial i e final j, da mesma forma que suas durações d_{ij}, com $d_{35}=d_{45}= 0$. Fonte: do autor.

O mesmo projeto P do parágrafo anterior, pode ser representado através da rede PDM, conforme pode-se visualizar na figura 2.4. As atividades $\{A, B, C, D, E, F\}$ passam a ser caracterizadas pelas nomenclaturas $\{A_2, A_3, A_4, A_5, A_6, A_7\}$,

respectivamente. As atividades fictícias A_1 e A_8, são inseridas no início e no final da rede, e as durações $\{d_2,..., d_7\}$ também passam a ser atributos dos nós do grafo.

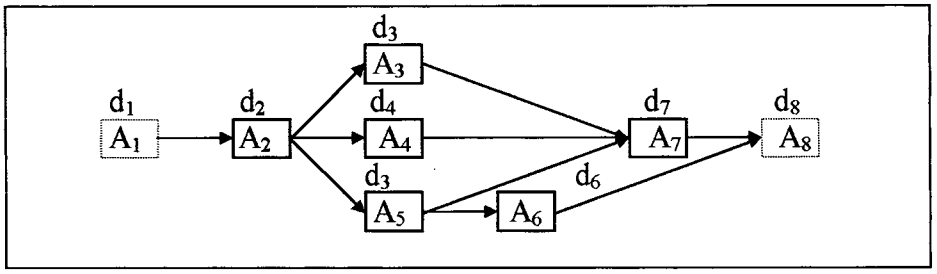


Fig. 2.4 - Exemplo de Rede PDM: as atividades $\{A_1, A_2, \ldots, A_8\}$ são interligadas por arcos e possuem as durações $\{d_1, d_2, \ldots, d_8\}$, com $d_1=d_8=0$. Fonte: do autor.

2.7. Os Problemas de Alocação de Recursos

Os tempos de início e de término resultantes da aplicação das redes CPM, PERT e PDM, implicam em perfis do uso de recursos ao longo do tempo de realização do projeto. Alocar um recurso significa designá-lo para uma determinada atividade por período de tempo, de forma a não acontecer de a capacidade de consumo ser maior que a disponibilidade. Em um processo de alocação de recursos, podem ocorrer duas situações: (1) a oferta é maior que a demanda, e neste caso, os recursos não são fatores limitantes na implementação do projeto, ou (2) a demanda é maior que a oferta em uma ou mais unidades de tempo.

Na segunda situação, podem ocorrer três tipos importantes de problemas: (1) haver necessidade de reduzir a variação nos perfis de demanda, (2) ser necessário reduzir a duração do projeto, e para isto, ter que adicionar recursos ao menor custo, e (3) ser necessário gerar uma combinação das datas de início das atividades, de modo que o recurso disponível não seja ultrapassado pelas despesas em nenhum período de tempo. Portanto, os problemas de alocação de recursos podem ser classificados em três tipos principais [Davis, 1973], respectivamente: o Problema do Nivelamento de Recursos, o Problema da Compressão de Projetos e o Problema da Alocação de Recursos Limitados. A formulação original do Gráfico de Gantt e dos métodos baseados em redes é orientada para a programação baseada no tempo, e não provêem um procedimento organizado de solução para estes três tipos de problemas.

2.7.1. O Problema do Nivelamento de Recursos (RLP - *Resouce Leveling Problem*)

O Problema do Nivelamento de Recursos surge quando há recursos em quantidade suficiente, mas por alguma razão torna-se necessário reduzir as flutuações do padrão de utilização. O objetivo da programação neste caso, é uniformizar o quanto possível os requerimentos de recursos; como não há limites de recursos, o processo de nivelamento consiste em manipular somente as atividades não críticas, utilizando suas folgas disponíveis. Agindo deste modo, sem manipular as atividades críticas, a duração do projeto permanece fixa.

Numerosos procedimentos heurísticos tem sido desenvolvidos para o nivelamento de recursos. A idéia básica destes procedimentos é produzir um histograma de recursos, gerado à partir da programação das atividades nas suas primeiras datas possíveis de início, e de acordo com um conjunto particular de regras, deslocar as atividades não críticas para unidades de tempo posteriores, até que o histograma esteja nivelado. Exemplos destas técnicas heurísticas podem ser encontradas nos trabalhos de Burges e Killebrev, (1962), Shaffer (1965), Antill (1982) e Harris (1990).

No tocante à otimização analítica, muitos trabalhos tem sido desenvolvidos, mas trazem o inconveniente de servir apenas às redes de pequeno tamanho. Entre eles, pode-se exemplificar o procedimento propostos por Ahuja (1984), que usa uma enumeração completa para as possíveis soluções; o proposto por Tavares (1987), que utiliza uma formulação baseada em programação dinâmica; os trabalhos de Easa (1989) e Mattila e Abraham (1998), que formulam o problema como um modelo de otimização linear inteira; e o procedimento de Bandelloni *et al.* (1994), que apresenta um modelo baseado em programação dinâmica não serial.

2.7.2. O Problema da Compressão de Projetos (TCTOP - *Time/Cost Trade-Off Problem*)

O Problema da Compressão de Projetos (TCTOP) surge quando a programação apresentada excede o prazo estabelecido para sua conclusão. Para resolvê-lo, verifica-se quais atividades pertinentes possuem um ou mais modos de produção de menor duração, reconhecendo que reduções de tempo implicam geralmente no aumento do custo. Feita a identificação, deve-se definir o modo conveniente para cada uma das atividades, e assim compor o conjunto das decisões que resulta na melhor relação tempo-custo para o empreendimento, satisfazendo-se ao prazo.

O TCTOP, também denominado Problema da Aceleração de Projetos, foi introduzido por Fulkerson (1961) e Kelley (1961), formulado como um problema de Programação Linear. A suposição básica destes trabalhos é a de que a função tempo-custo de uma atividade é composta de fragmentos lineares, convexos e contínuos, e isto implica em que as alternativas de duração-recurso para as atividades podem ser expressas como uma função de duração-custo contínua. Variações desta formulação são encontradas em Elmaghraby (1977) e Moder *et al.*, (1983).

O caso em que as relações tempo-custo são definidas como pontos discretos, representando alternativas distintas, é mais difícil de solucionar. No entanto, devido à predominância desta situação na prática [Harvey e Patterson, 1979; Hinderlang e Muth, 1979], vários trabalhos tem sido desenvolvidos. Meyer e Shaffer (1963), apresentam uma formulação baseada em programação inteira mista; Robinson (1975), apresenta uma formulação envolvendo programação dinâmica; Perera (1980), apresenta um modelo de programação inteira. Os métodos apresentados nestes trabalhos aplicam-se somente à problemas de pequeno porte; para problemas maiores, foram apresentados entre outros, os métodos de Feng e Burns (1997) com Algoritmos Genéticos, e de Adeli e Karim (1998) com Redes Neurais Artificiais.

2.7.3. O Problema de Programação de Projetos com Limitação de Recursos (RCPSP - *Resource-Constrained Project Scheduling Problem*)

O Problema da Programação de Projetos com Limitação de Recursos, é uma generalização do problema *job-shop* estático e portanto, pertencente à classe dos problemas de otimização NP-Completo [Parker e Rardin, 1981, 1982; Blazewincz *et al.*, 1983]. Ele enfoca a questão de como um conjunto de atividades inter-relacionadas tecnologicamente (*precedence constraints*), sob restrições de recursos (*resource constraints*), pode ser programado de forma a atender um objetivo da administração. A pesquisa intensa sobre procedimentos que resolvam este tipo de problema, surge do reconhecimento de que os modelos de rede CPM, PERT e PDM, pressupõem disponibilidade ilimitada de recursos [Davis, 1973].

A formulação do RCPSC clássico ou na forma padrão, pode ser observado na figura 2.5, sob as seguintes suposições [Demeulemeester e Herroelen, 1996]:

(a) Um projeto consiste de diferentes atividades, as quais são representadas no formato atividade-no-nó (*activity-on-the-node format* - um grafo direcionado e acíclico, no qual os nós representam as atividades e os arcos representam as restrições de precedência). Duas atividades fantasmas são introduzidas: a atividade *1* representa a atividade de início do projeto e é a predecessora direta ou indireta de toda atividade do projeto, enquanto a atividade N denota a atividade de final do projeto e é uma sucessora direta ou indireta de toda atividade.

(b) As atividades estão relacionadas por um jogo de relações de precedência Final-Início com um atraso de tempo zero, implicando que nenhuma atividade pode iniciar antes que as suas predecessoras tenham sido completadas.

(c) Nenhuma data de início ou de término é imposta a qualquer atividade do projeto.

(d) Cada atividade i $\{i = 1,...,N\}$ tem uma duração constante d_i (tempos de preparação são desprezíveis ou estão incluídas na duração fixada).

(e) Cada atividade i requer um número constante de unidades r_{ik} de um recurso renovável do tipo k ($k = 1,...,K$). Os requerimentos de recurso r_{ik} são constantes conhecidas sobre o intervalo de processamento da atividade.

(f) A disponibilidade a_k de um recurso renovável do tipo k é também uma constante conhecida ao longo do intervalo de duração do projeto.

(g) Nenhuma atividade pode ser interrompida depois de iniciada (não é permitido preempção de atividade).

(h) O objetivo é completar o projeto tão logo quanto possível, sem violar qualquer restrição de recurso e de precedência.

Problema da Programação de Projetos com Restrição de Recursos

Minimize f_N

Sujeito a

$$f_i \leq f_j - d_j \quad \forall (i, j) \in H,$$

$$f_1 = 0,$$

$$\sum_{i \in St} r_{ik} \leq a_k \quad \forall \quad k = 1,...,K; \quad t = 1,...,f_n,$$

Onde

a_k : *A disponibilidade total do recurso tipo k;*
d_i : *duração de atividade i;*
f_i : *data de término da atividade i;*
N : *número de atividades no projeto;*
H : *conjunto de pares de atividades indicando relações de precedência Final-início;*
K : *número de tipos de recurso;*
r_{ik}: *a quantidade de recursos do tipo k que é requerida pela atividade i;*
S_t : *Conjunto de atividades em progresso durante o intervalo de tempo* $(t-1, t] = \{i / f_i - d_i < t < f_i\}$;

Fig. 2.5 - O Problema da Programação de Projetos com Restrição de Recursos (*RCPSP- Resource Constrained Project Scheduling Problem*). Fonte: Demeulemeester e Herroelen, 1996.

2.8. Caracterização Geral do RCPSP

O Problema da Programação de Projetos com Restrição de Recursos pode ser caracterizado pelos seguintes itens [Lopez Vaca, 1995]: número de projetos simultâneos, natureza das informações do projeto, tipos de ligações permitidos; possibilidade de interrupção, modos de execução, tipos de recursos utilizados, número de recursos utilizados, número de objetivos e tipos de objetivos:

■ Número de Projetos Simultâneos

Um problema de programação de projetos pode envolver um único projeto (*single-project scheduling*) ou pode envolver vários projetos simultaneamente

(*multi-project scheduling*). Sobre o caso de múltiplos projetos, menos comum, tem-se entre outros os trabalhos de Pritsker *et al* (1969); Kurtulus e Narula (1982, 1985) e Mohanthy e Siddiq (1989).

■ Natureza das Informações do Projeto

Quando os dados sobre as atividades e os recursos envolvidos são determinados com precisão, diz-se que o problema é de natureza determinística; do contrário, quando alguns dos principais dados são variáveis segundo uma determinada distribuição de freqüências, diz-se que o problema é de natureza probabilística [Elmaghraby, 1990].

■ Tipos de Ligações Permitidos

Os problemas podem permitir um ou mais tipos de ligações entre atividades: Início-Início (*Start to Start*); Início-Final (*Start to Finish*); Final-Início (*Finish to Start*); e Final-Final (*Finish to Finish*) [Moder *et al.*, 1983].

■ Possibilidade de Interrupção

Na forma original do problema abordado, toda e qualquer atividade, uma vez iniciada, não pode ser interrompida (*nonpreemptive case*) [Boctor, 1990]; são mais freqüentes os artigos técnicos que apresentam procedimentos baseados nesta premissa. Métodos voltados para a possibilidade de interrupção das atividades (*preemptive case*) podem ser encontrados em Schrage (1972), Weglarz *et al.* (1977) e Slowinski (1980, 1981).

■ Modos de Execução

Quanto ao modo de execução, a forma clássica de abordagem envolve apenas uma e somente uma forma possível de execução para uma atividades (*single mode*). Quando existe mais de uma forma de execução para uma ou mais atividades, pode-se dizer que o problema pertence à classe de múltiplos modos (*multi-mode*). A grande maioria dos textos referem-se ao modo único; para o caso múltiplo, recomenda-se Boctor (1990,1996), Drexl e Gruenewald (1993) e Lopez Vaca (1995).

■ Tipos de Recursos Utilizados

Cada um dos recursos utilizados, pode ser classificado em uma das três categorias: Renovável, Não-renovável e Duplamente Restrito.

• Renovável (*Renewable*) - Quando o recurso em análise é limitado em quantidade, mas renovável de período à período. Ex: mão-de-obra, quando não há nova contratação durante a vida do projeto.

• Não-renovável (*Non-Renewable*) - Quando a soma do recurso analisado é limitada para o projeto como um todo, não havendo renovação por período. Ex: o orçamento de um projeto, quando em condições assim determinadas.

• Duplamente Restrito (*Doubly Constrained*) - Quando o recurso destinado ao projeto é limitado duplamente, na soma total e por período. Ex: o recurso financeiro disponível para a execução de um projeto.

■ **Número de Recursos Utilizados**

Um projeto pode utilizar apenas um único recurso (*single resource*), ou se possível, pode expressar todos os recursos envolvidos sob apenas uma denominação, como o recurso financeiro, por exemplo. Em ambos os casos, diz-se que o recurso é único. De outro lado, estão os problemas de múltiplos recursos (*multiple resources*), de natureza mais complexa, e presentes em artigos como Mohantly e Siddiq (1989);

■ **Número de Objetivos**

Um problema de RCPSP geralmente possui um único objetivo (*single objective*) a ser maximizado ou minimizado, como é o caso do procedimento padrão, que minimiza a duração total do projeto. No entanto, é crescente o número de metodologias que trabalham simultaneamente com objetivos múltiplos (*multiples objectives*), como em Davis *et al.* (1975), Kurtulus e Davis (1982), Epstein *et al.*, (1992), e Li (1996).

■ **Tipos de Objetivos**

Existem muitos tipos de objetivos possíveis para um RCPSP; os tipos mais comuns são:

• Minimização do Tempo de Execução do Projeto (*Project Completion Time*), utilizado em trabalhos como o de Brandt *et al.* (1964), Wiest (1967), Pritsker *et al.* (1969), Davis e Heidron (1971), Patterson e Huber (1974) e Talbot e Patterson (1978).

• Minimização do Custo Total do Projeto (*Overall Project Cost*), que pode ser visto em Phillips e Dessouki (1977), Slowinski (1980, 1981) e Talbot (1982).

• Maximização do Valor Presente do Projeto (*Project Present Value*), encontrado em Russel (1970); Grinold (1972); Doersch e Patterson (1977); Bey, Doersch e Patterson (1986); Elmaghraby e Herroelen (1990); Padman e Smith-Daniels (1993); Herroelen e Gallens (1993); Kazaz e Sepil (1996); Icmeli e Erenguc (1996) e outros.

2.9. A Literatura do RCPSP

O RCPSP foi introduzido por Kelley (1963) e Wiest (1963) [*in* Boctor, 1990]. A pesquisa sobre este tipo de problema pode ser classificada de acordo com dois

critérios: a função objetivo e as categorias de recursos (Ozdamar e Ulusoy, 1994). Em se tratando de função objetivo, os objetivos mais popularizados são a minimização da duração do projeto (objetivo baseado no tempo) e a maximização do valor presente do projeto (objetivo baseado no custo) [Doersch e Patterson, 1977; Yang, Talbot e Patterson, 1993].

A pesquisa envolvendo recursos renováveis e com o objetivo de minimizar a duração do projeto, inclui procedimentos de otimização tais como: (1) formulações baseadas em programação inteira [Pritsker *et al.*, 1969; Patterson e Huber, 1974; e Talbot e Patterson, 1978]; e (2) métodos *branch and bound* [Shrage, 1970; Hastings, 1972; Christofides *et al.*, 1987; e Bell e Park, 1990]. A abundância de variáveis inteiras e o número de restrições na formulação inteira, bem como a natureza exponencial do algoritmo *branch and bound*, tem impedido o desenvolvimento de procedimentos para problemas de tamanho grande.

Quando a utilização de recursos refere-se aos não renováveis, uma referência pode ser Slowinsky (1980, 1981), no qual as atividades são representadas por uma função de recursos com tempo discreto, que possibilita uma atividade ser realizada em um de seus possíveis modos de operação; Weglarz (1980, 1981), utiliza modelos com uma função tempo-recurso contínua, representando as atividades; e Talbot (1982), aborda problemas de grande porte.

Trabalhos com recursos duplamente restritos podem ser encontrados em Schrage (1972), Slovinky (1980, 1981) e Weglarz (1981).

Considerando que o problema é NP-Completo, os algoritmos não podem encontrar um ponto de ótimo ou até mesmo um boa solução para problemas grandes e complexos [Cho e Kim, 1997]. Para resolver eficientemente problemas de tamanhos realísticos, tem sido desenvolvidas heurísticas para os três tipos de recursos e ambos os objetivos. Para minimizar a duração do projeto e utilizando recursos renováveis cita-se como exemplo Norbis e Smith (1986, 1988), Alvarez-Valdes e Tamarit (1989), Ulusoy e Ozdamar (1989), Boctor (1990), Khattab e Choobineh (1990, 1991) e Bell e Hann (1991). Para minimizar a duração do projeto e utilizando recursos não renováveis, cita-se Li e Willis (1992).

A maioria das heurísticas existentes emprega regras de prioridades para programar as atividades, as quais permitem solucionar conflitos de recursos através da programação de atividades com mais alta prioridade, no tempo mais cedo de disponibilidade do recurso restrito. Mas, embora tenham sido elaboradas muitas regras de prioridade, é demonstrado que não há regras que dominem outras ou executem constantemente melhor que outras regras. Para conter tal deficiência de regras individuais, Boctor (1990) e Khattab e Choobineh (1991), sugerem composições de regras heurísticas que incluem várias regras de prioridade e selecionam a melhor solução entre regras individuais, como solução final.

Outra corrente de aproximações para o RCPSP, sugere a aplicação de procedimentos de melhoria para uma solução inicial possível. Yang *et al.*(1989), Bell e Han (1991) e Li e Willis (1992) desenvolveram heurísticas de multi-passo (*multi-pass heuristics*). Oguz e Bala (1994) sugerem um algoritmo de busca mais direto usando sub-

gradientes e técnicas de busca *n*-dimensionais. Estes métodos de busca páram quando encontram um mínimo local, e não conseguem escapar para um mínimo global.

Para superar tal desvantagem destes métodos de busca, tem sido frequentemente utilizados o *Simulated Annealing*, os Algoritmos Genéticos e o *Tabu Search*. Nesse campo, Icmeli e Erenguc (1994) apresentam um algoritmo *Tabu Search* no qual um movimento é feito para uma solução de vizinhança obtida pelo início de uma atividade em uma unidade de tempo mais cedo ou mais tarde. Mas, soluções não viáveis podem ser geradas com a aplicação deste método. Recentemente Lee e Kim (1996) desenvolveram uma solução simples que codifica um esquema em que uma possível solução de vizinhança pode ser sempre obtida de qualquer solução atual. Neste esquema, uma solução é representada por um vetor de números que denotam a prioridade das atividades, e com as prioridades definidas, a programação é obtida através de um método de programação por prioridade. Para encontrar uma boa combinação de atividades, os autores utilizaram o *Simulated Annealing*, os Algoritmos Genéticos e o *Tabu Search*.

Finalmente, uma revisão mais completa da literatura e a compreensão das heurísticas podem ser vistas em Davis e Patterson (1975) e Alvarez-Valdes e Valdes e Tamarit (1989).

2.10. Procedimentos para Solucionar o RCPSP

O RCPSP pertence à classe dos problemas combinatoriais, a qual é caracterizada pelo crescimento fatorial do tempo computacional requerido para considerar todas as possíveis soluções, de acordo com o aumento do tamanho do problema [Davis, 1973]. Portanto, os procedimentos para solucioná-lo podem ser extraídos da Otimização Combinatorial.

A Otimização Combinatorial (CO) estuda problemas que são caracterizados por um número finito de soluções possíveis. Embora em princípio, a solução ótima para um problema finito possa ser encontrada por simples enumeração, na prática esta tarefa é freqüentemente impossível, especialmente para problemas práticos de tamanho realístico onde o número de soluções possível pode ser extremamente alto [Papadimitriou e Steiglitz, 1985; Bjornal *et al.*, 1995]. Os pesquisadores de CO estudam as propriedades estruturais dos problemas e utilizam estas propriedades para criar técnicas de solução geral exatas e aproximadas.

Pode-se dividir os métodos de solução utilizados na Otimização Combinatorial, em dois grandes grupos: os procedimentos ótimos, que visam produzir a melhor solução através de programação matemática mais rigorosa, e os métodos de solução heurísticos, que visam produzir boas soluções:

■ Os Procedimentos Ótimos

Os procedimentos ótimos, também denominados exatos ou analíticos, subdividem-se em duas principais classes [MacCarthy e Liu, 1993]: os ótimos

eficientes, que geram programações ótimas em tempo polinomial; e os ótimos enumerativos, que consistem de enumerações do conjunto de soluções possíveis. Devido o fato de estes procedimentos serem inadequados para serem aplicados a problemas de tamanho e complexidade grandes, os mesmos não serão abordados neste trabalho.

■ Os Procedimentos Heurísticos

Em Pesquisa Operacional, o termo "Heurística" é usualmente entendido como sendo um algoritmo iterativo que não converge em direção à ótima solução, mas à uma solução possível do problema [Streim, 1975; Müller-Merbach, 1981]. Os mais importantes campos de ação da heurística são [Silver, Vidal e De Werra, 1980]:

- Problemas para os quais não existem algoritmos eficientes de convergência (*converging algorithms*), ou seja, algoritmos que não convergem para a solução em um tempo computacional aceitável.

- Problemas para os quais existem algoritmos eficientes de convergência, e que heurísticas podem ser utilizadas para acelerar o processo de solução. Como exemplo, o caso em que as heurísticas são utilizadas para encontrar uma boa solução inicial.

- Algoritmos de convergência que contém elementos de heurística, tal como a regra de seleção de um *pivot* em Programação Linear.

- Em diversas técnicas de busca em árvores como *Branch and Bound*, *bounded enumeration*, entre outros, como propostas *look-ahead*.

A maioria dos problemas para os quais não existem ou não podem ser desenvolvidos algoritmos de convergência eficientes, são de um tipo combinatorial (*combinatorial type*), no qual eles teriam que fazer arranjos, agrupamentos, ordenação ou seleção de objetos discretos.

Em resumo, os métodos de solução heurísticas para o RCPSP basicamente envolvem seis diferentes tipos [Storer, Wu e Vaccari, 1992; Kolisch, 1996]: (1) Programação baseada em regras de priorização de único e múltiplos passos; (2) Procedimentos *branch and bound* truncados [Alvarez-Valdes e Tamarit, 1989]; (3) Heurísticas baseadas em programação inteira [Oguz e Bala, 1994]; (4) Conceitos de arco disjuntivo [Shaffer *et al.*, 1965; Bell e Han 1991]; (5) Técnicas de busca local [Sampson e Weiss, 1993; Leon e Balakrishnan, 1995, Cho e Kim, 1997], e (6) os Algoritmos Evolucionários.

Devido a sua importância para este trabalho, somente as Heurísticas de Programação Baseadas em Regras de Prioridade, as heurísticas de Busca Local e os Algoritmos Evolucionários serão abordados nos próximos itens.

2.10.1. Heurísticas de Programação Baseadas em Regras de Prioridade

As técnicas de solução heurísticas baseadas em regras de prioridade ainda estão entre as mais importantes técnicas de programação para resolver o RCPSP [Kolisch, 1996], devido a três razões principais: (1) o método é intuitivo e fácil de usar, tornando altamente satisfatório seu uso em pacotes comerciais; (2) o método exige pouco esforço computacional, por isso é recomendável para ser integrado nos métodos de busca local da Inteligência Artificial [Storer *et al.*, 1992; Leon e Balakfishnan, 1995]; e (3) as implementações multi-passo do método apresentam alguns dos melhores resultados alcançáveis por heurística atualmente [Kolisch, 1995].

A maioria dos métodos heurísticos baseados em regras de prioridade, pode ser descrita através do procedimento descrito na figura 2.6 [Boctor,1993]. Para este procedimento, uma atividade é considerada programável se duas condições forem satisfeitas: todas as suas atividades predecessoras já estão programadas e se ela possui pelo menos um modo de execução possível. Um modo de execução só é considerado possível se cada recurso requerido é menor ou igual à soma do recurso disponível para o período de tempo considerado.

Heurística de Programação Baseada em Regra de Prioridade

Enquanto *Houver Alguma Atividade não Programada* ***Faça***
Avance para o Período mais Próximo onde há algum Recurso
[não Alocado;
Estabeleça a lista de atividades programáveis. ***Se*** *a lista*
[estiver vazia, volte para o passo anterior;
Escolha a atividade programável que possui a mais alta
[prioridade. Para esta atividade, escolha entre os
[possíveis modos duração-recurso, aquele que
[possui a maior prioridade.

Fig. 2.6 - Estrutura Básica em Pseudocódigo da Heurística de Programação Baseada em Regra de Prioridade. Fonte: Kolisch, 1996.

Geralmente, uma heurística de programação baseada em regras de prioridade, é composta de dois componentes: um esquema de geração de programação, e uma regra de prioridade.

Dois diferentes esquemas de programação podem ser distinguidos: o Método Serial (*Serial Method*) e o Método Paralelo (*Parallel Method*). Estudos comparativos entre ambos os métodos pode ser encontrados em Alvarez-Valdes e Tamarit (1989), Valls *et al.* (1992) e Kolisch (1996).

O Método Serial e o Método Paralelo descritos neste tópico, são relativos ao procedimento de único passo, conjuntamente com uma única regra de prioridade, os quais são empregados para encontrar uma solução possível. Os procedimentos de

múltiplos passos, ao contrário, realizam vários únicos passos em ordem para gerar uma amostra de possíveis soluções diferentes, onde a melhor é escolhida [Levy *et al.* (1962), Wiest (1967), Cooper (1976), Alvarez-Valdes e Tamarit (1989), Boctor (1990), Li e Willis (1992)].

■ O Método Serial

O Método Serial foi proposto por Kelley (1963); conforme pode-se observar na figura 2.7, a qual contém um exemplo de formulação, o método consiste de J estágios, em cada um dos quais uma atividade é selecionada e programada.

Esquema de Programação Serial

Inicialização

$n := 1$; $\{n = 1, \ldots, J$ *estágios*$\}$

$S_n := \phi$;

Enquanto $| S_n | < J$ ***faça*** *Estágio n*

Compute D_n *e* K'_{rt}, $\{t = 1, \ldots, T\}$ $\{r \in R\}$;

$j^* := mín_{j \in Dn} \{j / v(j) = inf_{i \in Dn} \{v(i)\}\}$;

$FC_j := máx \{F_i / i \in P_{j^*}\} + d_{j^*}$;

$F_j := mín \{t / FC_j^* \leq t \leq FT_{j^*}\}$, *com*

$\{k_{j^*r} \leq K'_{rI}\}$, $\{Ij = t - d_j^* + 1, \ldots, t\}$ *e* $\{r \in R\}$;

$S_{n+1} := S_n \cup \{j^*\}$;

$n := n+1$;

Onde:

d_j :*Duração da Atividade j;*

D_n :*Conjunto de Decisão;* $D_n := \{j / j \notin S_n, P_j \subseteq S_n\}$

F_j :*Data Programada de Término para a Atividade j;*

FC_j :*Data Mais Cedo Possível para o Término da Atividade j;*

FT_j :*Data Mais Tarde Possível para o Término da Atividade j;*

S_n :*Conjunto das Atividades Programadas;*

P_i :*Conjunto das Atividades Precedentes da Atividade j;*

$v(j)$:*Valor da Prioridade da Atividade j;* $\{j \in D_n\}$

I_j :*Data Programada de Início para a Atividade j;*

K'_{rt}:*Quantidade de Recurso k Disponível no tempo t;*

$$K'_{rt} := \sum_{j \in A_t} K_r - k_{jr}$$

A_t : *Conjunto das Atividades Programadas no Tempo t;* $A_t \subset S_n$

K_r :*Nível de Disponibilidade do Recurso r;*

k_{jr} :*Quantidade do Recurso r utilizado pela Atividade j por Tempo;*

Fig. 2.7 - Esquema da Programação Serial (*SSS - Serial Schedule Scheme*) em Pseudocódigo. Fonte: Kolisch, 1996.

Associado com cada estágio estão dois conjuntos de atividades disjuntos: No conjunto S_n estão as atividades programadas; O conjunto de decisão D_n contém as atividades não programadas, cujas predecessoras já encontram-se no conjunto S_n. Em cada estágio, uma atividade j do conjunto de decisão é selecionada através de uma regra de prioridade e programada para iniciar na data mais cedo Ij que satisfaça as restrições de precedências e de recursos; no caso de duas atividades possuírem mesma prioridade, será prioritária a atividade de menor número. Depois, a atividade selecionada é removida de D_n e colocada em S_n e as atividades que já possuem todas as precedentes já programadas, podem entrar em D_n. O algoritmo termina no estágio $n=J$, quando todas as atividades estiverem programadas.

Em uma rede do tipo Atividade-no-nó, FCj (Final Cedo) é a data mais cedo que uma atividade pode terminar, respeitando-se o cumprimento de todas as precedências; FTj (Final Tarde) é a data mais tarde que uma atividade pode terminar, de forma a não exceder o prazo T fixado para o término do projeto; Fj (Final) denota a data programada para o final da atividade, consistindo na menor data entre FCj e FTj, em que haja recurso disponível.

Trabalhos sobre o Método Serial em um ambiente de único passo, podem ser encontrados em Pascoe (1966), Cooper (1976, 1977), Boctor (1990) e Valls *et al.* (1992) e Kolisch (1996).

■ O Método Paralelo

Atualmente, dois algoritmos são associados com o Método Paralelo: o Algoritmo de Kelley (1963) e o Algoritmo de Brooks [Bedworth e Bayley, 1982]. Conforme pode-se observar na figura 2.8, este método consiste de J estágios, em cada um dos quais um conjunto de atividades é programado. A característica que diferencia o método paralelo, é que cada estágio n é associado com um tempo de programa t_n, onde $t_m \leq t_n$ para $m<n$. Por causa deste tempo de programa, o conjunto de atividades programadas é dividido em dois subconjuntos: o conjunto completo C_n , que contém as atividades que estavam programadas e estão completadas até o tempo de programa; e o conjunto ativo A_n, que contém as atividades que estavam programadas, mas que ainda estão ativas no tempo t_n. O conjunto de decisão D_n, diferentemente do que acontece no Método Serial, contém também todas as atividades não programadas que estão disponíveis para a programação, em termos de restrição de precedência e de recursos. O programa parcial de cada estágio é feito pelas atividades que estão no conjunto completo e no conjunto ativo. Cada estágio é feito em dois passos: (1) o tempo de programa novo é determinado e as atividades com um tempo final igual ao novo tempo de programa são removidas do conjunto ativo e colocadas no conjunto completo; (2) uma atividade do conjunto de decisão é selecionada com a regra de prioridade e programada para iniciar no tempo de programa corrente. Depois, esta atividade é removida do conjunto de decisão e colocada no conjunto ativo. O passo (2) é repetido até que o conjunto de decisão esteja vazio (as atividades estão programadas ou não existe mais recurso disponível para a programação). O Método Paralelo termina quando todas as atividades estão no conjunto ativo.

Experimentos computacionais utilizando a versão de único passo do Método Paralelo são reportadas por Pascoe (1966), Patterson (1973, 1976), Davis e Patterson (1975), Thesen (1976), Whitehouse e Brown (1979), Elsayed (1982), Alvarez-Valdes e Tamarit (1989), Ulusoy e Özdamar (1989), Boctor (1990), Valls *et al.* (1992). Implementações sobre o Método Paralelo podem ser vistos em Arora e Sachdeva (1989). Estudos sobre a performance das regras de priorização, podem ser encontradas em Alvarez-Valdes e Tamarit (1989) e Davis e Patterson (1975).

Esquema de Programação Paralela

Inicialização

$n := 1$; $\{n = 1, \ldots, J$ *estágios*$\}$
$t_n := 0$;
$D_n := \{1\}$;
$A_n := C_n := \phi$;
$K'_r := K_r \quad \forall\ r \in R$;
Vá Para *o Passo 2;*

Enquanto $|A_n \cup C_n| < J$ ***faça*** *Estágio n*

(1) $t_n := min\{F_j / j \in A_{n-1}\}$;
$A_n := A_{n-1} \mid \{j / j \in A_{n-1}, F_j = t_n\}$;
$C_n := C_{n-1} \cup \{j / j \in A_{n-1}, F_j = t_n\}$;
Compute $K'_r \quad \forall \quad r \in R \quad e\, D_n$;

(2) $j^* := min_{j \in Dn}\{j / v(j) = inf_{i \in Dn}\{v(i)\}\}$;
$F_{j^*} := t_n + d_{j^*}$;
$A_n := A_n \cup \{j^*\}$
Compute $K'_r \quad \forall \quad r \in R \quad e\, D_n$;
If $D_n \neq \phi$ ***Então Vá Para*** *o Passo 2* ***Senão*** $n := n + 1$;

Onde:

A_n *: Conjunto das Atividades Ativas no tempo* t_n;
C_n *:Conjunto das Atividades completadas até o tempo* t_n;
d_j *:Duração da Atividade j;*
D_n *:Conjunto de Decisão;*
$D_n := \{j / j \notin \{A_n \cup C_n\}, P_j \subseteq C_n, K'_{jr} <= K_r \ \forall \quad r \in R\}$
F *:Data Programada de Término para a Atividade j;*
P_i *:Conjunto das Atividades Precedentes da Atividade j;*
$v(j)$*:Valor da Prioridade da Atividade j;* $\{j \in D_n\}$
K'_{rt}*:Quantidade de Recurso k Disponível no tempo t;*

$$K'_{rt} := \sum_{j \in A_n} K_r - k_{jr}$$

K_r *:Nível de Disponibilidade do Recurso r;*
k_{jr} *:Quantidade do Recurso r utilizado pela Atividade j por Tempo;*

Fig. 2.8 - O Algoritmo de Brooks como Exemplo do Esquema da Programação Paralela (*PSS - Parallel Schedule Scheme*), em Pseudocódigo. Fonte: Kolisch, 1996.

2.10.1.1. Critérios de Priorização de Atividades

As principais regras de priorização de atividades, utilizadas nas heurísticas de solução do RCPSP são as seguintes [Boctor , 1993]:

- Folga Total (FT) mínima (*minimum total slack*);
- Última Data de Término (UDT) mínima (m*inimum late finish time*);
- Número máximo de Atividades Imediatamente Sucessoras (m*aximum number of immediate successors*)
- Tempo máximo de processamento (*maximum processing time*);
- Número máximo de Candidatas Subsequentes (*maximum number of subsequent candidates*).

2.10.1.2. Critérios de Priorização de Modos

As principais regras de priorização de modos de execução das atividades, em uma heurística de RCPSP são as seguintes [Boctor, 1993]:

- Menor modo possível (*shortest feasible mode*);
- Menor recurso crítico (*least critical resource*);
- Menor proporção de recurso (*least resource proportion*);

2.10.2. Busca Local (*Local Search* ou *Neighbourhood Search*)

Os métodos de busca local para otimização combinatorial, como o *Tabu Seach* [Glover *et al.*, 1985; Glover e Greenberg, 1989; De Werra e Hertz, 1989; Glover, 1986, 1989, 1990; Hertz e De Werra,1990; Hertz, 1994] e o *Simulated Annealing* [Van Laahoven e Aarts (1987); Eglese, 1990; Jeffcoat e Bulfin, 1993, Shtub *et al.*, 1996], realizam uma série de estágios melhorando a solução pelo movimento para uma vizinhança. Eles tem sido muito utilizados como um meio para encontrar soluções razoáveis para problemas combinatoriais pesados [Papadimitrou e Steiglitz, 1982].

O conceito de vizinhança, está baseado na possibilidade de poder ser especificado um método que possa perturbar uma solução encontrada, de modo à obter uma ou mais diferentes soluções para um problema. O conjunto das soluções que pode ser obtido desta forma, à partir de uma dada solução *s*, é chamado de vizinhança de *s*.

Os algoritmos de Busca Local sistematicamente investigam uma vizinhança particular e o algoritmo pára se nenhum vizinho conduz a uma melhoria de custo. Em geral, o melhor vizinho é aceito a cada iteração. A estrutura básica destes métodos pode ser observada na figura 2.9.

Método de Busca em Vizinhança Genérico

Inicialização
S := Vetor Solução Inicial
Enquanto a Condição de Parada não for Verdadeira
s' : = vetor vizinho de s;
Se f(s) - f(s') > 0 então s : = s';
Retorne s;

Fig. 2.9 - Estrutura Básica em Pseudocódigo, do Método de Busca em Vizinhança genérico. Fonte: Eglese, 1990.

Devido a dificuldade encontrada pelos métodos tradicionais para resolver problemas envolvendo restrições, há no meio científico, um grande interesse em métodos novos. Por este motivo, estas heurísticas tem sido muito pesquisadas recentemente. Neste contexto, Anderson (1996), Evans (1987) e Yannakakis (1990), apresentam um bom trabalho sobre a teoria, metodologia e análise estrutural das heurísticas de busca local. Os métodos de busca em vizinhança, também são comumente utilizados como uma base para algoritmos e soluções heurísticas nos problemas de otimização combinatorial, ou para melhorar uma solução que tenha sido gerada por uma outra heurística [Papadimitriou e Steiglitz, 1982].

2.10.3. Programação Evolucionária (*Evolutionary Programming*)

Nas décadas de 50 e 60, vários pesquisadores da ciência da computação estudaram os sistemas evolucionários naturais com a idéia de que a evolução poderia ser utilizada como uma ferramenta para resolver problemas de engenharia. Deste estudo surgiu a Programação Evolucionária, fundamentada no princípio de que os indivíduos mais adaptados ao meio ambiente, possuem maior probabilidade de sobrevivência.

Estes algoritmos pesquisam a melhor solução através da melhoria contínua de uma *população*, formada por soluções potenciais X^i (i=1,...,p) denominadas *indivíduos* ou *Cromossomas*. Cada evolução de uma população é denominada *Geração*, a qual consiste de um ciclos de: *Avaliação*, realizada por uma função objetivo; *Seleção*, baseada na determinação da *Aptidão* dos indivíduos; e *Reprodução*, que consiste da criação de *descendentes* através de *Recombinações* e *Mutações* de indivíduos selecionados. O esquema geral dos Algoritmos Evolucionários pode ser visto na figura 2.10.

Existem três classes principais de Algoritmos Evolucionários: a classe denominada Programação Evolucionária (*EP - Evolutionary Programming*) desenvolvidos inicialmente por Fogel, Owens e Walsh [*in* Goldberg, 1989; Davis, 1991; Pierreval e Tautou, 1996; Jacobs, 1996; Mitchel, 1996]; a classe Estratégias Evolucionária (*ES - Evolutionary Strategies*), desenvolvida por Rochenberg e Schwefel (1965, 1973) [*Apud* Goldberg, 1989; Davis, 1991; Pierreval e Tautou, 1996; Jacobs,

1996; Mitchel, 1996]; e a classe Algoritmos Genéticos, introduzida por Holland em 1975, a qual constitui o assunto do capítulo III deste trabalho.

Estrutura dos Algoritmos Evolucionários

(1) Defina a Codificação Apropriada para as Soluções;
(2) Gere uma População Inicial $P(0)=X^1, X^2,..., X^p$*; Faça* $t=0$*;*
(2) Avaliação: Compute a Aptidão $f(X^i)$ *para cada indivíduo* X^i, $i=1,...,p$*;*
(4) Seleção: Selecione Indivíduos de S(t); Isto resulta no grupo de acasalamento S(t). O Mesmo Indivíduo de P(t) pode Aparecer Várias Vezes em S(t).
(5) Reprodução: Formar pares de indivíduos em S(t) e para cada par de indivíduos:
- *Recombine o par com probabilidade* p_{cross} *e copie o descendente resultante em S(t+1); O par de cromossomas é eliminado e substituído pelo seu descendente;*
- *Copie os pares de indivíduos no conjunto S(t+1) com probabilidade* $1-p_{cross}$.

Para Cada Indivíduo de S(t+1):
- *Aplique a Mutação no Indivíduo com Probabilidade* p_{mut} *e copie os Indivíduos Transformados em P(t+1);*
- *Copie o Indivíduo em P(t+1) com Probabilidade* $1-p_{mut}$ *;*

(5) Faça t:=t+1 e Retorne para o Passo 2 até que o Critério de Parada seja Atingido.

Fig. 2.10 - Estrutura Básica dos Algoritmos Evolucionários. Fonte: Pierreval e Tautou, 1996.

2.11. Os Projetos Lineares de Construção Civil

Na Construção Civil, denominam-se Projetos Lineares àqueles caracterizados por um número significativo de atividades repetitivas [Arditi e Albulak, 1979; Selinger, 1980; Russel, 1988; Mosheli, 1993]. Como exemplo, pode-se citar os projetos relativos às seguintes obras: rodovias, ferrovias, canais ou redes de tubulações, edifícios isolados (se existe um pavimento-tipo), conjuntos de edifícios ou casas (cada bloco ou casa constitui uma unidade repetitiva), e pontes (repetição de fundações, pilares, vigas e lajes). Construção de Massa (*Mass Construction*, O'Brien, 1969) é outra denominação utilizada para este tipo de construção, por força da analogia com a linha de produção da indústria fabril, posto que a construção de seus componentes produzidos mais significantes, como algum elemento estrutural que se repete em vários pavimentos, comporta-se como uma linha de produção em série. Na terminologia da área, cada unidade repetitiva é também denominada estágio ou seção.

2.12. O Uso das Técnicas de Rede na Programação de Projetos Lineares

Segundo Stradal e Cacha (1982) e Reda (1990), o uso das técnicas de rede em projetos lineares possui duas grandes desvantagens: primeiro, elas requerem um

grande número de atividades (vide fig. 2.11) para representar o projeto, tornando difícil a visualização global e exigindo uma grande capacidade computacional para processar os dados; segundo, as técnicas de rede não garantem a continuidade do trabalho. Por isto, Selinger (1980), reporta de suas experiências, que para obter a máxima utilização de recursos ao modelar um projeto repetitivo com uso de redes, foi necessário impor regras para garantir a continuidade do trabalho; e isto sem dúvida torna menos eficiente o trabalho destas ferramentas.

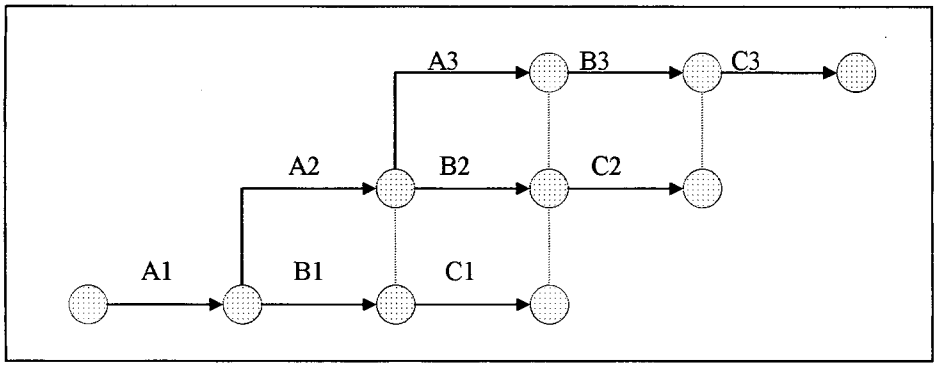


Fig. 2.11 - Representação de um Projeto Repetitivo na Rede CPM: as Atividades A1, A2 e A3, Referem-se à mesma Atividade A, quando Realizada nas Seções 1, 2 e 3. Idem para as Atividades B e C. Fonte: do autor.

Em outras palavras, as redes são mais apropriadas para a programação de produtos cujos componentes de um mesmo tipo são processados uma única vez. Nos projetos lineares, uma atividade repetitiva é análoga a um componente que é processado um número de vezes que corresponde ao número de seções.

2.13. Métodos de Sequenciamento de Trabalho

De acordo com Stradal e Cacha (1982), em geral, existem três tipos de sequenciamento de trabalho (fig. 2.12): o Método do Trabalho Simultâneo, o Método do Trabalho Sucessivo e o Método da Linha de Fluxo.

No primeiro tipo, o mesmo trabalho é realizado simultaneamente em todas as seções. Por este motivo, este é o procedimento mais rápido, mas exige que todas as seções estejam prontas para iniciar o trabalho, bem como causa um grande consumo de recursos; sua utilização pode ser viável quando for necessário obter o máximo de ganho de tempo devido a atrasos de projeto, quando uma atividade possui muitas sucessoras, ou quando sua finalização rápida é estratégica. O Método do Trabalho Sucessivo realiza um conjunto de atividades em uma seção, antes de iniciar na seguinte, por isto é o mais longo dos métodos de seqüenciamento de trabalho. Sua utilização é recomendável quando é necessária a conclusão de uma seção para iniciar outra. Finalmente, o método da linha de fluxo impõe um fluxo sem interrupção e um ritmo constante a cada seção, para cada equipe que realiza um trabalho repetitivo. É o método mais recomendado devido a sua eficiência.

Na figura 2.12, encontra-se a expressão gráfica dos três métodos. Cada atividade *a*, *b* e *c*, é representada por um tipo gráfico diferente, e as seções repetitivas são simbolizadas por números romanos. Em 2.12.A pode-se observar o Método de Trabalho Simultâneo, em 2.12.B encontra-se o Método de Trabalho Sucessivo e em 2.12.C e 2.12..D visualiza-se o Método da Linha de Fluxo.

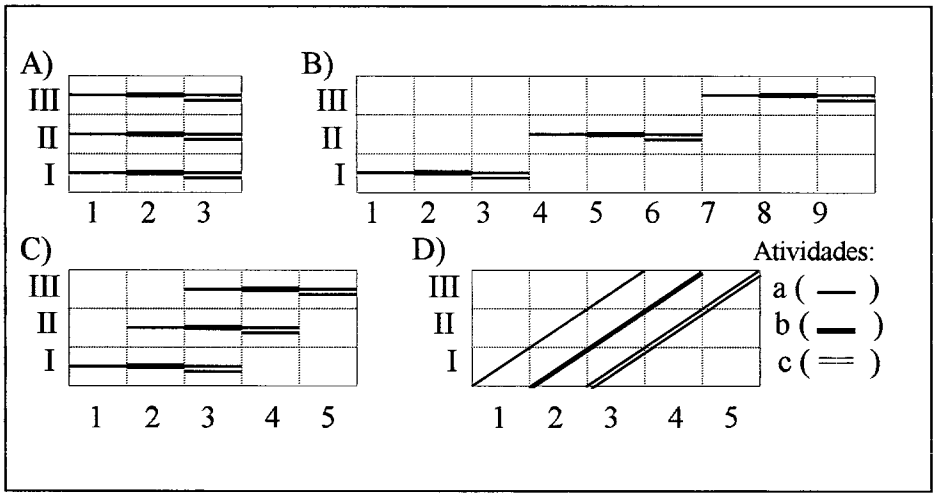


Fig. 2.12 - Tipos de Seqüenciamento de Trabalho: (A) O Método de Trabalho Simultâneo, (B) O Método de Trabalho Sucessivo e (C) e (D) O Método da Linha de Fluxo. Em cada quadro, estão presentes as seções I, II, III, as atividades *a*, *b* e *c*, e as unidades de tempo {1,2...}. Fonte: Stradal e Cacha, 1982.

2.14. Características dos Agrupamentos de Fluxos

Um grupo de fluxos pode possuir as seguintes características [Popescu, 1979; Stradal e Cacha, 1982], observadas na figura 2.13: (A) o grupo é ritmado e

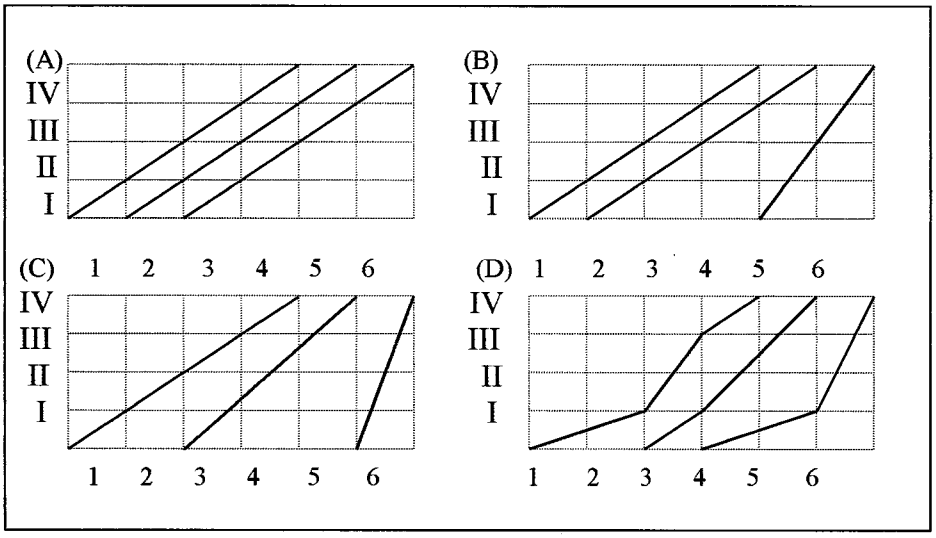


Fig. 2.13 - Métodos de Linha de Fluxo: (A) fluxos ritmados e sincronizados; (B) fluxos ritmados e não sincronizados, com exceção de poucos fluxos; (C) Fluxos ritmados e não sincronizados; (D) Fluxos não ritmados e não sincronizados. No eixo *y* estão as seções I, II, III, e IV; e no eixo *x* estão as unidades de tempo 1,...,6. Fonte: Stradal e Cacha, 1982.

sincronizado, ou seja, todo fluxo individual possui a mesma velocidade em cada seção e o conjunto dos fluxos possui uma velocidade única; (B) o grupo é ritmado e sincronizado, mas possui algumas poucas atividades não sincronizadas (como é o caso da atividade mais à direita); (C) o grupo é ritmado e não sincronizado, isto é, nenhuma atividade é sincronizada; e (D) o grupo é não ritmado e não sincronizado.

2.15. O Desenvolvimento dos Métodos de Programação de Projetos Lineares

As desvantagens amplamente reconhecidas em usar o tradicional gráfico de barras e técnicas de rede para planejar projetos de construção lineares conduziram ao desenvolvimento de uma família de técnicas de planejamento de projetos lineares, que permite a administração efetiva e a sincronização dos recursos necessários para as atividades repetitivas.

A origem destas técnicas não é clara, e segundo Arditi (1986), pode ter acontecido simultaneamente em vários países, com o intuito de resolver problemas de produção industrial. O que se pode afirmar, é que elas vem sendo desenvolvidas desde a década de 50 [Suhail e Neale, 1994], sob os mesmos princípios orientados para os recursos [Kleinfeld, 1976; Birrel, 1980]. Dentre as mais importantes, destacam-se as seguintes: o Método da Linha de Balanço (*Line of Balance Method*) [Lumsden, 1968; Oxley e Poskitt, 1971; Carr e Meyer, 1974; Levine *et al.*, 1976; Harris e McCafferr, 1977; Arditi e Albulak, 1979; Arditi e Albulak, 1986, Al Sarraj, 1990]; o VPM - *Vertical Production Method* [O'Brien, 1975, 1984, 1985]; o *Time Space Scheduling* [Johnston, 1981; Stradal e Cacha, 1982]; o *Cascade Networks* [Rist, 1972]; o *Velocity Diagrams* [Roech, 1972]; o *Time Velocity Diagrams* [Dressler, 1974]; o *Fenced Bar-Charts* [Melin, 1981, 1984]; o *Chain Bar-Charts* [Perera, 1981, 1982]; o *Construction Management System* [Anderson *et al.*, 1968; Pedersen, 1972]; o *Combined PERT/LOB* [Schoderbek e Digman, 1967; Harbert, 1976]; o *Flow Charts* [Forbes, 1971]; o *Flow Line Method* [Cole, 1977]; o LSM - *Linear Scheduling Method* [Chrazanowsky e Johnston, 1986]; o RPM - *Repetitive Project Model* [Reda, 1990]; e o *Time Chainage Charts* [Mawdesley *et al.*, 1990].

Johnston (1981), sumarizou o desenvolvimento dos Métodos de Programação de Projetos Lineares na indústria da construção; Lutz e Hijazi (1993) apresentaram uma revisão das técnicas existentes para planejar e analisar operações de construções lineares. A literatura indica que este métodos podem ser usados em projetos tais como: esquemas residenciais múltiplos [Roech, 1972; Carr e Meyer,1974; Ashley, 1980; Perera, 1981, 1983]; pontes [Selinger, 1980], edifícios [Anderson e Fjosne, 1968; Pedersen, 1972; O'Brien, 1975; Mangin, 1979, Walker, 1979], oleodutos [Dressler, 1974; Levine, 1976] e estradas [Arditi e Albulak, 1979, 1986; Johnston, 1981].

Dentre os trabalhos que fazem uma abordagem de modelação matemático/computacional nos últimos trinta anos, pode-se citar no campo da Programação Linear: Perera, 1983, Handa e Barcia (1986); Reda, 1990; e Eldin e Senouci, 1994. A Programação dinâmica foi introduzida por Selinger (1980); Mosheli (1993) incorpora a variável custo, ao solucionar um Problema de Aceleração de Projetos. Todos estes trabalhos, de Programação Linear e Programação Dinâmica, envolvem taxas de produção fixas, não admitem atividades paralelas e as atividades possuem duração constante em cada unidade repetitiva. Estas limitações foram

superadas por Senouci e Eldin (1996), através de uma formulação dinâmica para o Problema da Compressão de Projetos, direcionada a projetos seriais e não seriais. Outros pesquisadores utilizaram técnicas de simulação, como Ashley (1980), e métodos estocásticos, como Dressler (1974). A possibilidade de interromper a continuidade do trabalho das atividades repetitivas, pode ser encontrada na formulação de Russel e Caselton (1988).

No campo da Inteligência Artificial, tem sido desenvolvidos sistemas especialistas baseados no conhecimento (*KBESs: Knowledge-Based Expert Systems*), que usam informações geométricas deduzidas de modelos computacionais tridimensionais, e incorporam conhecimentos a respeito das necessidades e restrições para a geração automática de planos e programas: Zozaya-Gorostiza *et al.*, (1990) com o *Construction Planex*, Chang e Ibbs (1990) com o *Priority Ranking*, Morad e Beliveau (1993) com o *Know-Plan*, Zouen e Tommelien (1993), com o *MovePlan*. Thabet (1992) [*Apud* Thabet e Beliveau, 1997] e Thabet e Beliveau (1993, 1994 e 1997), descrevem um procedimento estruturado para incorporar restrições horizontais e verticais na programação de atividades repetitivas de edifícios, denominado *HVLS* (*Horizontal and Vertical Logic Scheduling*), o qual é associado com outros métodos de programação para desenvolver o *SCaRC* (*Space-Constrained and Resource Constrained*), um sistema especialista para a programação de pavimentos repetitivos em edifícios, bem como para desenvolver o *HISCHED* [Shaked e Warszawski, 1995].

2.16. O Método da Linha de Balanço (*LOB - Line of Balance Method*)

O Método da Linha de Balanço é um dos métodos mais conhecidos entre os pesquisadores para a programação de projetos lineares. Sua origem é derivada da indústria de manufatura na década de 40 [Neale e Neale, 1989], e foi posteriormente desenvolvida pelo *U.S. Navy Department* nos anos 50. Seu objetivo foi o de atingir ou avaliar a taxa de fluxo de produtos acabados em uma linha de produção [Al Sarraj, 1990].

O LOB é uma ferramenta orientada para a programação de recursos [Trimble, 1984], e graficamente revela divergências entre o programa e o progresso real das atividades, habilitando a administração à avaliação quantitativa [Khristy, 1970]. Lumsden (1968) modificou a técnica básica desta ferramenta e a aplicou na programação de construção residencial; em seu trabalho, encontram-se os princípios do método, sua representação e análise. Khisty (1970) a aplicou na produção e suprimento de concreto e em obras de reparo de um porto; Carr e Meyer (1974) descreveram o LOB na sua forma atual; Arditi e Albulak (1986), descrevem um experimento de uso em projetos de construção de estradas; Sarraj (1990), apresentam uma formalização matemática para o método, preenchendo uma lacuna na literatura.

A figura 2.14 mostra dois exemplos do emprego do Método da Linha de Balanço, construídos a partir de uma rede CPM. O eixo vertical contém a seqüência das unidades repetitivas $\{I, II, III\}$ a serem executadas; o eixo horizontal é relativo às unidades de tempo t, onde estão o tempo T (prazo de execução do projeto) e o ponto t_I (término das atividades na unidade repetitiva I); o angulo ω é a razão de produção do projeto. O primeiro gráfico é ritmado, e constitui a chamada Programação Paralela; o

segundo gráfico apresenta cada atividade com seu ritmo natural, e consiste na chamada Programação por Recursos.

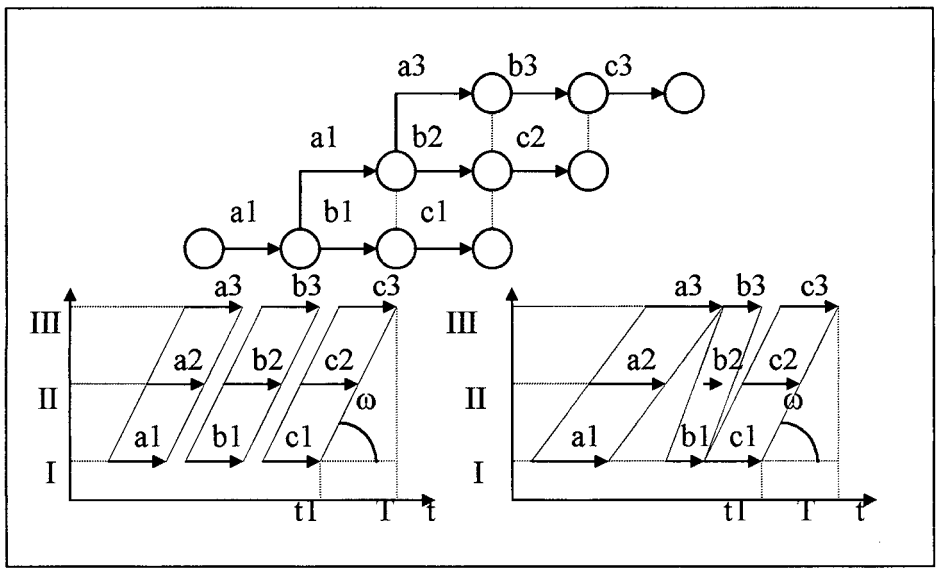


Fig. 2.14 - O Método da Linha de Balanço (*LOB:Line of Balance Method*), na forma ritmada (Programação Paralela) e na forma não ritmada (Programação por Recursos). Fonte: do autor.

2.17. A Curva de Agregação de Recursos

A Curva de Agregação de Recursos é uma técnica apropriada para analisar a distribuição de um tipo qualquer de recurso (ou de um conjunto de recursos) utilizado no projeto, em função do tempo. Em termos gráficos, esta técnica consiste na poligonal formada com base nos pontos médios de cada barra do histograma do recurso analisado. Quando esta ferramenta é expressada cumulativamente, recebe a denominação de Curva "S". Estas representações podem ser observadas na figura 2.15.

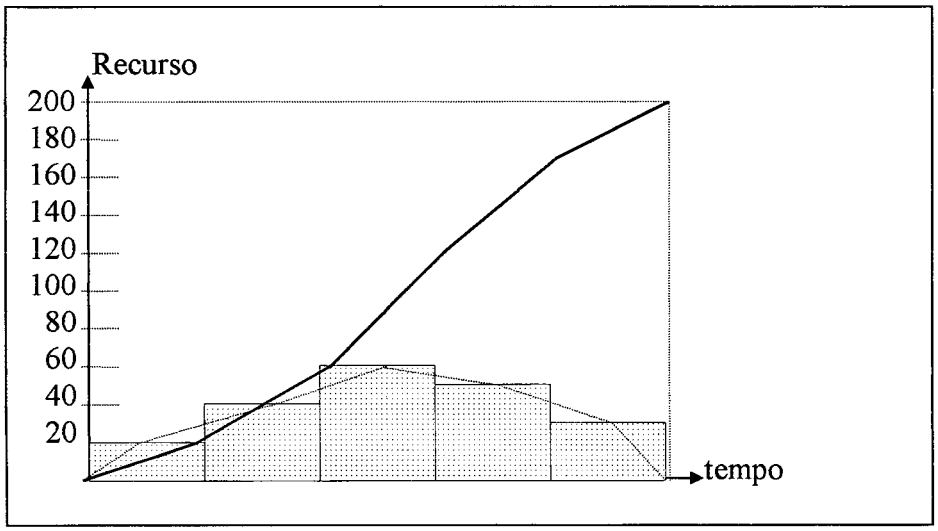


Fig. 2.15 - Exemplo de Histograma de Recursos (hachurado), Curva de Agregação de Recursos (tracejada) e Curva de Agregação de Recursos Acumulados (linha cheia) ou Curva "S". Fonte: do autor.

Dentre a grande quantidade de trabalhos publicados sobre o assunto, pode-se citar Archibaldi (1976) e Shtub *et al.* (1994), na indústria fabril; na construção Civil, os trabalhos de Perry (1970); Carr *et al.* (1974); Gates e Scarpa (1976); Peer (1982); Christian e Kallouris (1990); Heineck (1989, 1990), Kaka e Price (1993); Dawood (1994) e Casaroto (1995). Com base nos princípios desta ferramenta, é possível gerar modelos de planejamento do consumo dos recursos, bem como modelos de planejamento das despesas diretas dos projetos, fundamentados em dados históricos das empresas construtoras ou em curvas teóricas. Na figura 2.16 sobre as curvas "S", observa-se a curva *1*, obtida segundo a programação na data de início mais cedo e a curva *4*, obtida através da programação na data de início mais tarde; estas duas curvas constituem os limites superior e inferior de uma região que contém uma família de curvas, de onde sobressaem àquelas cujos pontos de inflexão resultam de combinações entre os valores 40, 50 e 60 (na figura, encontram-se as curvas 50-50 e 60-60).

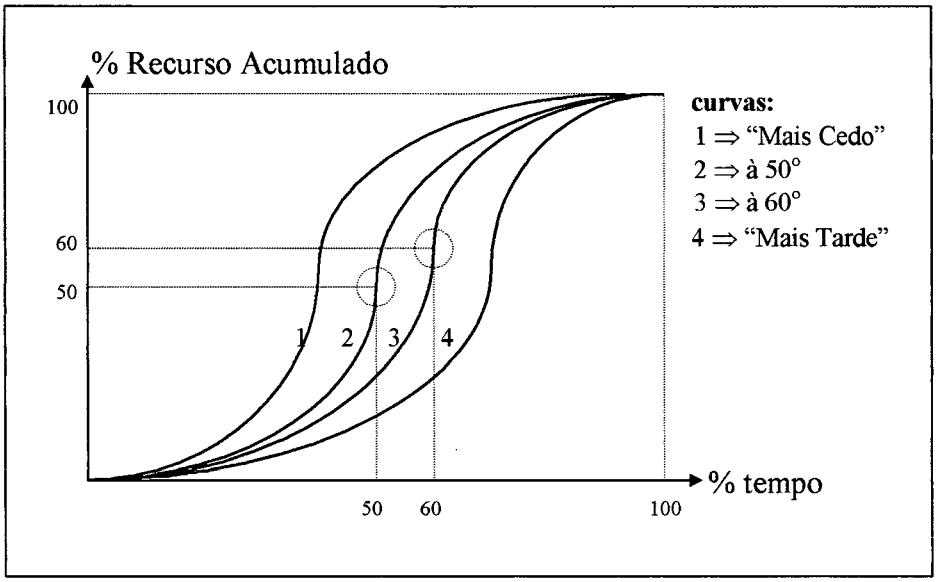


Fig. 2.16 - Esboço de Alguns Padrões teóricos da Curva "S": (1) programação segundo a data mais cedo, (2) programação à 50%; (3) programação à 60% e (4) programação segundo a última data de início. Fonte: do autor.

2.18. Conclusão

Quando se deseja modelar um projeto, na maioria das aplicações utiliza-se um grafo dirigido em um dos dois formatos: *activity-on-the-node* ou *activit-on-the-arrow*. Os métodos heurísticos de solução destinados à criar programações em função do tempo, mais tradicionalmente o Gráfico de Gantt, o CPM e o PDM, tem sua sustentação teórica fundamentada nesta analogia do projeto com um grafo.

Os problemas mais complexos de programação, os quais envolvem alocação de recursos, consistem nos problemas de nivelamento, nos problemas de compressão e nos problemas de alocação de recursos limitados. O Gráfico de Gantt e os métodos de rede originalmente não possuem mecanismos para resolver esta classe de

problemas, e embora ao longo do tempo tenham sido criados muitos procedimentos auxiliares, os mesmos apresentam limitações de eficiência e efetividade.

Os métodos de solução analíticos ou exatos utilizados para solucionar estes problemas, bem como os métodos enumerativos, aplicam-se somente à pequenos problemas e em um nível de abstração tal, que geralmente não podem ser aplicados à situações do mundo real. Portanto, os métodos heurísticos constituem o objeto mais importante da pesquisa nesta área, e na procura de métodos que tenham qualidades como robustez e eficiência, destacam-se os procedimentos de Busca Local, tal como os Algoritmos Genéticos, o *Tabu Search* e o *Simulated Annealing*.

O Problema da Programação de Projetos torna-se mais complexo quando refere-se às construções repetitivas. Para este tipo de construção, as ferramentas mais apropriadas são os Métodos de Programação Linear (MLP), dentre os quais pode-se citar o Método da Linha de Balanço. A orientação da programação voltada para os recursos do projeto, bem como a promoção da continuidade do trabalho das equipes nas seções repetitivas, figuram entre as vantagens proporcionadas pelos MLP.

A Curva de Agregação de Recursos constitui uma ferramenta gráfica importante para a análise de projetos. Sua aplicação não se restringe somente ao acompanhamento da obra, mas também na determinação de uma estratégia de distribuição dos recursos ao longo do projeto, extraída das políticas de fluxo de caixa da empresa e dos dados estatísticos de obras anteriores.

CAPÍTULO III

ALGORITMOS GENÉTICOS

3.1. Introdução

Neste capítulo está contida uma revisão teórica sobre o método de busca utilizado na solução do problema abordado nesta tese. Os Algoritmos Genéticos (AG's) referem-se a uma classe de procedimentos de busca adaptativa, inspirada nos princípios evolutivos das populações genéticas da natureza, e foram concebidos por Holland (*Michigan University*) nos anos 60. Sua denominação originou-se da analogia entre a representação de uma estrutura complexa através de um vetor de componentes e a idéia da estrutura genética de um cromossoma.

Em ambos os contextos, o natural e o dos Algoritmos Genéticos, dois conceitos são fundamentais: (1) a *evolução*, que pode ser conceituada como a geração expontânea e contínua de estruturas funcionais inovativas [Bedau e Packard, 1992], e (2) a *adaptação*, que designa qualquer processo através do qual uma estrutura é progressivamente modificada para obter melhor desempenho no seu ambiente [Holland, 1992]. Durante o curso da evolução, as populações naturais evoluem conforme os princípios de seleção natural e sobrevivência da adaptação. Ou seja, indivíduos que adaptam-se melhor ao ambiente, possuem maiores chances de sobreviver e reproduzir, enquanto os indivíduos de menor ajuste tendem a ser eliminados. Isto significa que os genes dos indivíduos altamente aptos influenciarão um número crescente de indivíduos a cada geração, e sua espécie evolui para se tornar cada vez mais adaptada ao seu ambiente.

Um AG simula este processo, tomando uma população inicial de indivíduos e aplicando operadores genéticos artificiais em cada geração. Em condições de otimização, cada indivíduo da população é codificado em um *string* ou cromossoma, o qual representa uma solução possível para um determinado problema, enquanto a adaptação dos indivíduos é avaliada através de uma função de aptidão. Basicamente, aos indivíduos altamente aptos (melhores soluções) são dadas maiores oportunidades de reproduzirem-se trocando partes de informação genética, em um procedimento de acasalamento denominado Cruzamento; o operador de Mutação é utilizado para alterar alguns genes nos cromossomas, e causar diversidade na população. A descendência ou nova população, pode substituir toda a população atual ou substituir apenas os indivíduos de menor ajuste. Este ciclo de avaliação, seleção e geração, é repetido até que uma solução satisfatória seja encontrada.

3.2. A Literatura dos Algoritmos Genéticos

O desenvolvimento inicial desta técnica ocorreu nas décadas de 60 e 70, e a primeira descrição rigorosa foi feita por Holland no livro *Adaptation in Natural and*

Artificial Systems (1975), no qual o autor também demonstrou o sucesso da aplicação em uma variedade de problemas, tais como reconhecimento de padrões, sistemas de classificação, configuração de redes de comunicação, sistemas de controle industrial e otimização combinatorial em geral. Ainda em termos de literatura básica no ano de 1975, De Jong completou sua tese de doutorado *An Analysis of the Behavior of a Class of Genetic Adaptative Systems*, combinando a teoria dos *Schemata* de Holland e suas experiências computacionais; Liepins e Hilliard (1989), apresentaram uma boa revisão dos maiores desenvolvimentos de aplicação; Goldberg (1989), Beasley *et al.* (1993) e Reeves (1993) fazem avaliações inclusivas e demonstram o conhecimento básico dos Algoritmos Genéticos; finalmente, uma abordagem mostrando dificuldades de aplicações em algumas áreas, pode ser encontrada em Reeves (1991).

As primeiras aplicações para problemas de otimização matemática bem definidos, foram apresentados por Hollstien (1971) [*in* Mitchell, 1996], em um sistema de controle computacional; seguido por De Jong (1975) [*in* Goldberg, 1989], com uma análise do comportamento de uma classe de sistemas adaptativos genéticos.

No campo da Pesquisa Operacional relacionado ao modelamento através de estruturas de grafos, Goldberg e Lingle Jr. (1985), Grefenstette *et al.* (1985), Whitley *et al.* (1991) apresentaram importantes trabalhos ligados ao Problema do Caixeiro Viajante (*Travelling Salesman Problem*) e Kapsalis *et al.* (1993) desenvolveram um AG para o Problema da Árvore de Steiner. Comparações interessantes sobre as performances de AG's, *Simulated Annealing* e um algoritmo de vizinhança simples, podem ser encontradas em Reeves (1995).

Os recentes avanços teóricos tem permitido o uso dos AG's nas indústrias. Smith (1983) e Schaffer (1985) utilizaram estes algoritmos no aprendizado de máquinas; Goldberg (1984) utilizou no controle de sistemas de operações em oleodutos; e Fourman (1985) resolveu um problema de *Layout* VLSI.

Em se tratando dos problemas de Programação da Produção, destacam-se os trabalhos de Davis (1985) e Biegal e Davern (1990) nos problemas *Job Shop*; Cleveland e Smith (1989), Gauthier (1993); Venkateswara (1993) e Neppalli *et al.* (1996) no problema de programação *Flow Shop*; Venugopal e Narendran (1992) propõem um modelo matemático bi-criterial para o Problema da Formação de Células (*Cell Formation Problem*) em Sistemas de Manufatura Celular; Levitin e Rubinovitz (1993) sugerem uma aplicação para resolver os casos *Linear Assignment Problem* (LAP) e *Ciclic Assignment Problem* (CAP) do *Quadratic Assignment Problem*; e López Vaca (1995) apresenta um algoritmo para a programação de projetos com minimização do *makespan* e nivelamento de recursos.

Nas pesquisas sobre Cruzamento (*Crossover*), o mais importante dos operadores genéticos, tem sido particularmente interessante o estudo das redundâncias e das informações perdidas que podem resultar das permutações feitas pelo operador utilizado. Alguns esquemas foram desenvolvidos para evitar as soluções inviáveis, tais como o Operador OX (*Order Crossover*) [Davis, 1991], o Operador PMX (*Partially Mapped Crossover*) [Goldberg, 1989] e o Operador CX (*Cycle Crossover*) [Goldberg,

1989]. Em outra linha de inovação, Beasley e Chu (1996) defendem uma taxa variável de mutação e propõem um novo operador de cruzamento (fusão), baseado no *fitness*.

Finalmente, é interessante observar que a literatura técnica dos AG's não prediz a performance desta técnica em relação à outros procedimentos heurísticos de forma conclusiva. As análises experimentais e teóricas tem sido geralmente dirigidas para a solução de um problema específico.

3.3. Quando é Interessante Utilizar um Algoritmo Genético

Não existe um consenso rigoroso sobre o campo de aplicabilidade dos Algoritmos Genéticos, mas muitos pesquisadores concordam em alguns pontos. Os AG's seriam interessantes para uma determinada aplicação se: o espaço a ser pesquisado é grande, o espaço a ser pesquisado não é *liso* ou unimodal, o espaço a ser pesquisado não é bem entendido, e o problema não requer solução ótima.

- **O Espaço a Ser Pesquisado é Grande**

Se o espaço a ser pesquisado for grande, não é interessante pesquisá-lo exaustivamente até que seja encontrado um ponto ótimo global;

- **O Espaço a ser Pesquisado não é *Liso* ou Unimodal**

Se o espaço é formado por uma simples colina lisa, um algoritmo gradiente-ascendente é mais eficiente que um algoritmo genético;

- **O Espaço a ser Pesquisado não é bem Entendido**

Se o espaço de pesquisa for bem entendido, os métodos de pesquisa que usam heurísticas de domínio específico podem ser utilizadas com a mesma desenvoltura de um algoritmo genético;

- **O problema não requer solução ótima**

O usuário deve ficar satisfeito com uma boa solução e ser beneficiado pela rapidez com que esta solução é encontrada, já que os AG's podem ou não encontrar um ótimo global.

3.4. Vantagens dos Algoritmos Genéticos

Os principais atrativos dos Algoritmos Genéticos reconhecidos na literatura técnica são: a independência do domínio, a não linearidade, a robustez, a facilidade de modificação e a sua natureza paralela.

- **Independência de Domínio**

Os AG's trabalham sobre a codificação do problema, de tal forma que é possível elaborar um programa computacional geral para resolver muitos problemas de otimização diferentes;

- **Não Linearidade**

Enquanto as técnicas de otimização convencionais se baseiam em suposições irrealistas de linearidade, convexidade e diferencialidade, entre outras, os AG's não utilizam nenhuma dessas suposições;

- **Robustez**

Como conseqüência da independência de domínio e da não linearidade, os AG's podem resolver uma diversidade de tipos de problemas, bem como podem trabalhar com funções não-lineares;

- **Facilidade de Modificação**

As modificações em um AG a fim de modelar variações do problema original são muito fáceis de ser implementadas, diferentemente de muitas outras heurísticas;

- **Natureza Paralela**

Um AG possui natureza tal que facilita a efetivação de implementações com processamento paralelo. Por exemplo, assumindo substituição em bloco, a aptidão de cada cromossoma poderia ser calculada em paralelo, reduzindo o tempo de processamento de cada geração.

3.5. A Robustez dos Algoritmos Genéticos

Além dos argumentos apresentados no item anterior, existem outros argumentos relativos à robustez dos Algoritmos Genéticos em relação aos métodos tradicionais de otimização:

- Os Algoritmos Genéticos são probabilísticos;
- A propriedade do paralelismo intrínseco, através da qual os AG's não melhoram apenas única solução, mas permitem a simultaneidade da avaliação das soluções;

- A diversidade genética, isto é, a cada passo do processo de busca, um conjunto de soluções candidatas é considerado e vários elementos são simultaneamente envolvidos na geração de novas candidatas;

- Os AG's testam o ajuste dos candidatos; ou seja, o ajuste não é necessariamente derivado de uma função objetivo a ser otimizada;

- O uso da codificação de um conjunto de parâmetros, em lugar dos próprios parâmetros;

- Um AG constitui um método de busca aleatória, portanto funciona melhor que heurísticas comuns quando se dispõe de pouca informação sobre o espaço de busca.

3.6. Terminologia dos Algoritmos Genéticos

A terminologia utilizada pelos Algoritmos Genéticos consiste basicamente de analogias extraídas da Biologia. Os termos mais utilizados são [Mitchel, 1996]:

- **Aptidão (*Fitness*)**

Probabilidade que um organismo possui para reproduzir (viabilidade), ou uma função do número de descendentes (fertilidade) que este organismo possui;

- **Cromossoma (*Chromosome*)**

Uma estrutura de solução candidata para o problema;

- **População (*Population*)**

Conjunto dos cromossomas que compõe cada geração;

- **Diplóide (*Diploid*)**

População cujos cromossomas são formados em pares;

- **Haplóide (*Haploid*)**

População cujos cromossomas não estão dispostos em pares;

- **Gene (*Gene*)**

Divisão conceitual ou bloco funcional de um cromossoma, capaz de codificar uma característica;

- **Posição (*Locus*)**

Posição em que um gene se localiza no cromossoma;

- **Alelo (*Allele*)**

Característica ou valor numérico que representa um gene. Como exemplo, um cromossoma que utiliza codificação binária, possui em cada posição (*locus*) dois possíveis alelos: *0* e *1*;

- **Genótipo (*Genotype*)**

Refere-se ao jogo particular de genes contido em um genoma (coleção completa de material genético de um organismo, envolvendo todos os cromossomas);

- **Cruzamento (*Crossover*)**

Troca de partes entre dois cromossomas, geralmente haplóides;

- **Gameta (*Gamete*)**

Cada cromossoma gerado por cruzamento;

- **Mutação (*Mutation*)**

Mudança ou troca de um ou mais alelos de um cromossoma;

3.7. Representação da Estrutura de Solução

De acordo com a Biologia, Cromossomas são extensas cadeias de compostos químicos que contém a descrição da composição genética dos seres vivos. Do ponto de vista dos AG's, o cromossoma é uma solução candidata, ou seja, um ponto no espaço de busca; cada cromossoma codifica uma solução do problema e seu valor contribui para o cálculo da função objetivo.

Em muitas aplicações e originalmente, um cromossoma consiste de um *string* de codificação binária (vide figura 3.1). A presença de um dos caracteres do código binário em uma determinada posição significa a existência ou não de uma determinada característica. Mas, esta codificação aparentemente simples não é única e existem muitas formas para implementá-la nos AG's; destacam-se neste sentido os trabalhos de Holland (1975), Caruana e Schafer (1988) e Goldberg (1989).

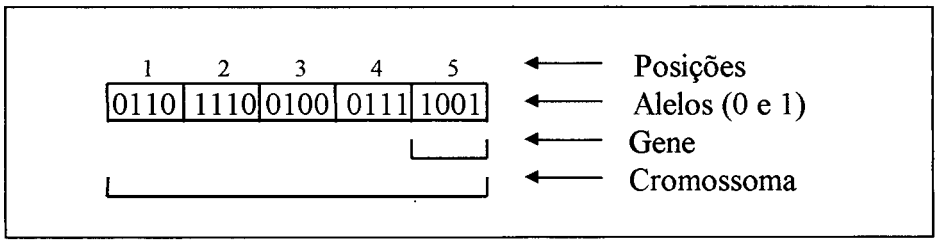


Fig. 3.1 - Estrutura de um Cromossoma em Codificação Binária. Fonte: do autor.

As soluções candidatas também podem ser implementadas em codificações não binárias, com o objetivo de melhorar a adaptação ao tipo de problema abordado. No entanto, as extensões teóricas existentes para codificações não binárias ainda não foram tão bem resolvidas como a codificação original. Estudos tem sido feitos em Janikow e Michalewicz (1991) e O'Reilly e Oppacher (1995).

3.8. Esquema de Cromossomas

A idéia de esquema permite referir-se de forma compacta às similaridades entre Cromossomas; um esquema (*schema*; plural: *schemata)* é uma estrutura de cromossoma com posições de conteúdo fixo e posições em aberto, tradicionalmente representadas por asteriscos. Sua finalidade é estabelecer "famílias" de cromossomas, caracterizadas originalmente pelos códigos binários imutáveis das posições fixas e pelo comprimento (número de genes) do cromossoma. Os melhores esquemas tendem a perpetuarem-se através das gerações e conseqüentemente também tende a perpetuar-se a sua contribuição para a função objetivo.

Utilizando como exemplo um cromossoma de tamanho *7*, pode-se visualizar que um esquema *1*001*1* possui quatro cromossomas {*1000111*, *1100111*, *1000101*, *1100101*}, o que leva a concluir que para alfabetos de cardinalidade K e *strings* de comprimento δ existem $(K+1)^{\delta}$ esquemas, bem como pode-se visualizar que para uma população de N indivíduos podem existir até $N*2^{\delta}$ esquemas, quando se considera que cada *string* também é um membro de 2^{δ} (considerando que já se tem o cromossoma). Os fundamentos matemáticos a respeito de esquemas podem ser encontrados em Holland (1975), Goldberg (1989) e Reeves (1993).

3.9. Avaliação das Soluções Candidatas

A Aptidão (*fitness*) refere-se ao grau com que uma solução candidata contribui para a convergência do algoritmo na busca da melhor solução. Para mensurar

esta grandeza utiliza-se uma função de aptidão, cujo objetivo fica sendo portanto estabelecer uma medida de qualidade para o cromossoma. Como exemplo, no caso em que o cromossoma utiliza codificação binária, o cálculo do *fitness* poderia consistir em fazer a decodificação do *string* para um número real e então avaliar a função objetivo para aquele ponto.

Na definição do processo de avaliação, surgem inúmeros problemas que exigem mudanças na função objetivo. Os principais são os seguintes:

- **Transformação da Função de Custo para Função de Lucro ou Utilidade**

Na maioria dos procedimentos comuns de otimização, basta multiplicar a função de custo por *-1*. No caso dos Algoritmos Genéticos, esta operação é insuficiente porque não oferece garantia sobre a não negatividade dos valores resultantes. Sendo o coeficiente C_{max} o maior valor da função objetivo até a geração atual, ou somente da população atual ou de um número determinado de últimas gerações, a função comumente usada é:

$$f(x) = \begin{cases} C_{max} - x, & \text{se } x < C_{max} \\ 0, & \text{em caso contrário} \end{cases}$$

Onde x = Valor da Função Objetivo ;
C_{max} = Maior Valor de f (x);

- **Resultados Negativos na Função de Custo ou Utilidade**

Neste tipo de função não existe dificuldade com a meta a ser atingida com o uso da função, posto que é suficiente a maximização. No entanto, pode-se ter problemas com o aparecimento de resultados negativos. Por este motivo, pode-se utilizar a seguinte transformação:

$$f(x) = \begin{cases} x + C_{min}, & \text{se } x = C_{min} > 0 \\ 0, & \text{em caso contrário} \end{cases}$$

Onde x = Valor da Função Objetivo ;
C_{min} = Menor Valor de f (x);

■ **Convergência Prematura**

A dominância dos cromossomas com alto valor de aptidão nas populações iniciais, pode fazer com que o algoritmo venha a convergir muito rapidamente para um ponto de máximo local no espaço de soluções. Deste fato, surge a necessidade de modificar a função objetivo em prol da contribuição dos indivíduos menos aptos, através de vários mecanismos, dentre os quais pode-se citar o procedimento de escala, o procedimento de *rank* ou o procedimento de torneio (os dois últimos serão mostrados no item referente à Seleção). A idéia básica do procedimento de escala é limitar a competição entre os cromossomas nas iterações iniciais e estimulá-la progressivamente, através de uma mudança (linear em geral) de escala na aptidão. No caso da mudança linear de escala, mostrado a seguir, os valores a e b são obtidos das condições $f_{médio} = Y_{médio}$ e $f_{máx} = k\ Y_{médio}$; o parâmetro k é o número esperado de descendentes do melhor cromossoma, usado para assegurar que o membro mais ajustado da população será escolhido k vezes na média.

$$f = a * Y + b$$

Onde: f = função de aptidão modificada;
Y = valor da função de Aptidão;
a e b = constantes.

■ **Atendimento às Restrições do Problema**

Neste caso, a função *fitness* expressa o quanto uma solução candidata é boa em termos de satisfação das restrições. Geralmente nos Algoritmos Genéticos recorre-se à Função *Penalty*, que permite representar de 70 a 80 % dos casos práticos [Chambers, 1995]. Neste caso, a violação de cada restrição é penalizada com um determinado peso não negativo, definido segundo a orientação dos métodos clássicos de resolução. O resultado da função consiste na soma das penalizações violadas:

$$f(x) = \sum_{k=1}^{K} [w_{ikjk} * d_{ikjk}(x)]$$

$$\text{com} \quad d_{ikjk}(x) = \begin{cases} 1, \text{ se } x \text{ viola a restrição } C_{ikjk} \\ 0, \text{ em caso contrário} \end{cases}$$

onde w_{ikjk} = peso relativo à restrição C_{ikjk};

3.10. Operadores Genéticos

Os operadores genéticos são instrumentos de variação dos AG's, possuindo a habilidade de causar variação nos esquemas existentes. Analisando sob o prisma da matemática, são operadores cuja função é criar novos pontos de busca no

espaço de solução, com base nos elementos da população atual. Existem inúmeros operadores para a manipulação dos cromossomas, e os mais comuns são s Seleção (*Selection*), o Cruzamento (*Crossover*), a Mutação (*Mutation*) e a Inversão (*Inversion*).

3.10.1. Seleção

O Operador de Seleção dos Algoritmos Genéticos é uma analogia com a seleção biológica natural de Darwin. Realizar uma seleção significa escolher dentre os indivíduos pertencentes à população, àqueles que servirão para criar descendentes para a próxima geração e quantos descendentes cada um deverá gerar. Uma seleção pode ser denominada *muito fraca*, se resultar em evolução muito baixa e *muito forte* se resultar em populações de indivíduos mais fortes (o termo *forte* aqui utilizado, refere-se aos indivíduos mais adaptados ao meio ambiente; esta adaptação é que direciona o processo evolutivo). Isto significa que os indivíduos mais adequados vivem mais e geram mais descendentes, os quais herdam suas qualidades. Fica portanto evidente, que o mecanismo da Seleção tem por finalidade enfatizar um filtrador de indivíduos na população, para aumentar a perspectiva de gerar indivíduos melhores segundo algum critério.

3.10.1.1. Mecanismos de Seleção

Um AG funciona com base na manutenção de uma população de cromossomas, que são os pais potenciais. A escolha dos pais pode ser feita de inúmeras formas e dentre os mecanismos mais importantes, figuram a Seleção Através da Proporcionalidade do *Fitness*, a Seleção Boltzmann, a Escalada Sigma e a Seleção *Rank*.

- **Seleção Através da Proporcionalidade do *Fitness* (*fitness-Proportionate Selection*)**

Esta denominação cabe a todo processo no qual o valor esperado do indivíduo é resultante da razão entre o seu *fitness* e o *fitness* médio da população. O método mais comum para realizar este operador é a Seleção pelo Giro da Roleta (*Roulette Wheel Selection*). Neste tipo de seleção análogo ao giro de uma roleta, denominada por Brindle (1981) [*Apud* Goldberg, 1989] de Amostragem Probabilística com Substituição (*Stochastic Sampling with Replacement*), cada cromossoma possui a probabilidade de ser escolhido, proporcionalmente à sua avaliação (valorada pela função de aptidão). Portanto, cada indivíduo possui a probabilidade $Psel=\delta_i/\Sigma\delta_i$ de ser escolhido, onde δ_i é a sua aptidão; ou seja, cada indivíduo da população possui um número de *fatias* da roleta proporcional à sua adaptação.

Segundo Mitchel (1996), quando se utiliza um processo de seleção por proporcionalidade do *fitness*, a convergência tende a ser prematura, visto que é dada muita ênfase desde as primeiras iterações, na exploração de indivíduos de *fitness* mais altos, dispensando outras regiões do espaço de busca. Posteriormente, quando existe muita semelhança entre os indivíduos da população, não existem diferenças de *fitness* que sejam capazes de continuar a evolução. Ao poder que faz com que os indivíduos

com maior *fitness* tenham mais descendentes, denomina-se pressão de seleção no vocabulário dos Algoritmos Genéticos.

A conclusão é que a taxa de evolução depende da variância do *fitness* da população. Neste sentido, diversos pesquisadores tem proposto outros tipos de seleção, que transformem os valores *brutos* dos *fitnesses* em valores esperados, para evitar a convergência prematura. Assim é que surgiram entre outros, operadores como a Seleção Boltzmann, a Escalada Sigma e a Seleção *Rank*:

■ Seleção Boltzmann (*Boltzmann Selection*)

O método de seleção Boltzmann [Goldberg, 1990; Prügel-Bennett e Shapiro, 1994], extraído do *Simulated Annealing*, estabelece uma pressão de seleção diferente à cada tempo da pesquisa de solução. Inicialmente, admite a reprodução de indivíduos com *fitness* baixo, permitindo assim manter a diversidade da população e evitar convergências prematuras, e posteriormente aumenta a pressão para enfatizar os indivíduos com *fitness* alto. Segundo a analogia estabelecida, a contínua variação da *temperatura* controla a taxa de seleção: quando a temperatura é alta, a pressão de seleção é baixa (todos os indivíduos possuem a mesma probabilidade de reprodução); a temperatura é gradualmente reduzida, incrementando a pressão de seleção, forçando que o algoritmo limite-se à melhor parte do espaço de pesquisa, mantendo um grau apropriado de diversidade. Na expressão abaixo, verifica-se que quando a temperatura T decresce, a diferença no valor esperado $VE(i,t)$ entre um *fitness* $f(i)$ baixo e outro alto é aumentada.

$$VE(i,t) = \frac{e^{f(i)/T}}{[e^{f(i)/T}]_t}$$

Onde VE (i, t) = Valor Esperado do indivíduo *i* no tempo *t*;
T = Temperatura;
$[e^{f(i)/T}]_t$ = Média sobre a população *i* no tempo *t*;

■ Escalada Sigma (*Sigma Scaling*)

Este operador mantém a pressão de seleção quase constante durante o processo de busca da melhor solução, sem depender da variância do *fitness* da população. O Escalada Sigma ou Truncamento Sigma (*Sigma Truncation*, em Goldberg, 1989) coloca o valor esperado em função de três grandezas: o seu *fitness*, a média da população e o desvio-padrão da população. Um exemplo deste operador pode ser encontrado em Mitchel, 1996:

$$VE(i,t) = \begin{cases} \dfrac{1 + f(i) - f'(t)}{2\,\sigma(t)}, & \text{se } \sigma(t) \neq 0; \\ 1, & \text{se } \sigma(t) = 0; \end{cases}$$

Onde VE (i, t) = Valor Esperado do indivíduo *i* no tempo *t*;
$f(i)$ = *Fitness* do indivíduo *i*;
$f'(t)$ = *Fitness* médio da população no tempo *t*;
$\sigma(t)$ = Desvio padrão do *fitness* da população no tempo *t*;

■ **Seleção Rank (*Rank Selection*)**

Analogamente aos operadores mostrados anteriormente, o principal propósito deste operador é evitar a convergência muito rápida. Segundo a versão linear proposta por Baker, 1985 [*Apud* Mitchell, 1996], os indivíduos da população são ordenados crescentemente de *0* a *N* de acordo com o seu *fitness* e o valor esperado depende desta ordem e não do valor absoluto do *fitness*. O usuário escolhe o valor esperado *Max* (*Max≥0*) do indivíduo com ordem *N*, com *1≤Max≤2*, *Min=2-Max* e $\sum_i VE(i,t)=N$; após este passo, o valor esperado de cada indivíduo da população no tempo *t* é definido por:

$$VE(i,t) = Min + (Max - Min)\frac{rank(i,t) - 1}{N - 1}$$

Onde VE = Valor Esperado do indivíduo *i* no tempo *t*;
Min = Valor esperado do indivíduo de ordem 1;
Max = Valor esperado do indivíduo de ordem N;

3.10.1.2. O Processo de Substituição dos Indivíduos

O processo de substituição dos indivíduos de uma população pode ser classificado em dois tipos principais: Substituição Tradicional e Substituição em Estado Estacionário.

■ **Substituição Tradicional**

Denomina-se substituição tradicional, ao processo de substituição total de todos os Cromossomas de uma população pelos seus descendentes, a fim de compor a nova população.

■ **Substituição em Estado Estacionário**

O tipo de reprodução dita em estado estacionário [Davis, 1989], prevê a substituição gradativa dos Cromossomas de uma população atual. À cada iteração do algoritmo, um ou mais Cromossomas são escolhidos para dar lugar aos descendentes. Este mecanismo foi elaborado a partir da necessidade de corrigir algumas distorções de outros operadores, como a destruição ou alteração de muitas características interessantes de uma população, quando seus elementos são submetidos aos operadores de Cruzamento e Mutação; bem como foi elaborado para evitar a exclusão de muitos dos bons indivíduos que não se reproduzem, quando submetidos à Reprodução Tradicional.

Quando o processo só aceita a substituição se o candidato descendente for diferente dos Cromossomas da população atual, a técnica recebe a denominação especial de Reprodução em Estado Estacionário Sem Duplicação.

3.10.2. Cruzamento

Muitos autores consideram que este operador é a principal particularidade dos Algoritmos Genéticos frente às outras heurísticas. Depois que dois cromossomas forem selecionados, o Cruzamento pode ser realizado se sua taxa ou probabilidade pré-definida de realização for atingida.

O Cruzamento consiste no operador de acasalamento, que permite a produção de novos *strings* através da troca de informações parciais entre pares de cromossomas, e embora existam outras formas de Cruzamento em desenvolvimento e experimentação, nesta revisão serão abordadas somente três: a de um único ponto de corte, a de dois pontos (incluindo variações) e uma que não utiliza pontos de corte (Operador CX). Quanto ao número de pais, serão mostrados somente operadores que utilizam dois pais; a utilização de um número maior para gerar descendentes por Cruzamento, constitui a classe de Operadores Multi-pais e de acordo com Chambers (1995), a idéia é marcar posições dos pais e depois passar estas características para os descendentes.

O Cruzamento de um único ponto é feito através da escolha aleatória de um só ponto de cruzamento. Cada par de cromossomas escolhidos como *pais*, gera dois descendentes através da permuta de suas partes finais, depois do ponto de cruzamento. A idéia é recombinar esquemas em diferentes *strings*. Os três principais problemas deste tipo de operador, são: (1) ele não pode combinar todos os esquemas possíveis; (2) esquemas de grande tamanho são sujeitos à destruição; em outras palavras, a criação ou destruição de um esquema depende fortemente da localização do *bit* no cromossoma [Eshelman, Caruana e Schaffer, 1989] e (3) os descendentes de dois *pais*, sempre contém os pontos finais dos *strings*, caracterizando uma tendência de sobrevivência das partes finais.

Para exemplificar o Cruzamento de um único ponto, os seguintes Cromossomas denominados *S1* e *S2*, a partir de uma posição gerada randomicamente

igual à *4*, darão origem aos descendentes *S1'* e *S2'*, conforme a ilustração apresentada a seguir:

S1 = 0 - 1 - 1 - [1 - 0 - 0 - 1 - 0 - 1

S2 = 1 - 0 - 1 - [0 - 1 - 1 - 1 - 1 - 0

A nova configuração seria:

S1' = 0 - 1 - 1 - [0 - 1 - 1 - 1 - 1 - 0

S2' = 1 - 0 - 1 - [1 - 0 - 0 - 1 - 0 - 1

O Cruzamento de Dois Pontos surgiu da necessidade de reduzir as carências do anterior. Duas posições são escolhidas randomicamente e os segmentos entre elas são trocados. Desta forma, fica menos provável a destruição de esquemas grandes, bem como os segmentos trocados nem sempre contém os pontos extremos dos *strings*. Utilizando os cromossomas *S1* e *S2*, e segundo duas posições aleatórias *3* e *6*, obtêm-se as configurações *S1'* e *S2'* mostradas a seguir:

S1 = 0 - 1 - [1 - 1 - 0 - 0]- 1 - 0 - 1

S2 = 1 - 0 - [1 - 0 - 1 - 1]- 1 - 1 – 0

A nova configuração seria:

S1' = 0 - 1 - [1 - 0 - 1 - 1]- 1 - 0 - 1

S2' = 1 - 0 - [1 - 1 - 0 - 0]- 1 - 1 - 0

Em problemas como o Problema do Caixeiro Viajante (*Traveling Salesman Problem*) e o Problema da Programação de Projetos (*Project Scheduling Problem*), nos quais o modelamento é feito através de grafos conectados sem ciclos e circuitos, a codificação do cromossoma é comumente realizada através de números inteiros, que representam as cidades e as atividades, respectivamente. O resultado é que os dois tipos de Cruzamento anteriormente mostrados, não são adequados devido às repetições que resultam da operação. Inúmeros operadores foram criados para contornar esta situação, e neste trabalho apresenta-se três tipos: o operador OX (*Order Crossover*), o operador PMX (*Partially Mapped Crossover*), o operador CX (*Cycle*

Crossover) e o operador LOX (*Linear Order Crossover*), ilustrados a seguir em exemplos simples.

■ **Operador PMX**

Considerando-se os seguintes *strings S1* e *S2*, inicialmente dois pontos de corte são escolhidos ao acaso, e os genes presentes entre os pontos de corte [*4 - 7 - 5*] e [*6 - 2 - 8*] são trocados de forma que ambos recebem partes de seqüência de informações genéticas novas.

S1 = 3 - 2 - 8 - [4 - 7 - 5] - 9 - 1 - 6

S2 = 4 - 9 - 5 - [6 - 2 - 8] - 3 - 7 - 1

A nova configuração seria:

S1' = 3 - 2 - 8 - [6 - 2 - 8] - 9 - 1 - 6

S2' = 4 - 9 - 5 - [4 - 7 - 5] - 3 - 7 - 1

Considerando que estas duas representações são ilegais devido a repetição de genes, fica definido o passo final que é substituir em cada cromossoma os elementos repetidos, mantendo a mesma seqüência entre os pontos de cruzamento:

S1' = 3 - 5 - 7 - [6 - 2 - 8] - 9 - 1 - 4

S2' = 8 - 9 - 2 - [4 - 7 - 5] - 3 - 6 – 1

■ **O Operador OX**

O Operador OX inicia de forma semelhante ao PMX, com a escolha fortuita de dois pontos de corte e transferência integral dos genes internos entre os pais. Utilizando os mesmos *strings S1* e *S2* do exemplo anterior, o procedimento consiste em transferir a seqüência anterior ao primeiro ponto de corte, para o final do *string*:

S1 = 3 - 5 - 7 - [6 - 2 - 8]- 9 - 1 - 4 - {*3 - 5 - 7*}

S2 = 8 - 9 - 2 - [4 - 7 - 5]- 3 - 6 - 1 - {*8 - 9 - 2*}

A seguir, troca-se a seqüência anterior ao primeiro ponto de corte de um pai, pela seqüência entre os pontos de corte do outro pai. No exemplo, a seqüência [*3-5-7*] de *S1*, foi trocada pela seqüência [4-7-5] de *S2*, e a seqüência [*8-9-2*] de *S2*, foi trocada pela seqüência [*6-2-8*] de *S1*. Marca-se em negrito os genes repetidos após o segundo ponto de corte:

S1 = 4 - 7 - 5 - [6 - 2 - 8] - 9 - 1 - **4** - {*3* - ***5*** - *7*}

S2 = 6 - 2 - 8 - [4 - 7 - 5] - 3 - **6** - 1 - {***8*** - *9* - ***2***}

Finalmente, elimina-se os genes repetidos marcados:

S1' = 4 - 7 - 5 - [6 - 2 - 8] - 9 - 1 - 3

S2' = 6 - 2 - 8 - [4 - 7 - 5] - 3 - 1 - 9

■ O Operador CX

O operador CX possui um procedimento muito diferente dos operadores PMX e OX, visto que não utiliza pontos de corte. Ele executa a recombinação de modo que cada gene das descendências vem das posições correspondentes de qualquer um dos dois pais. Usando novamente os *strings S1* e *S2*, começa-se da esquerda e escolhe-se um gene do primeiro pai. O filho *S1'* leva o valor do primeiro gene de *S1*:

S1' = 3 - ? - ? - ? - ? - ? - ? - ? - ?

O gene de mesma posição em *S2* possui valor *4*. Procurando-se este valor em *S1*, verifica-se que ele se encontra coincidentemente no quarto gene, e este valor deve ser levado para a quarta posição de *S1'*:

S1' = 3 - ? - ? - 4 - ? - ? - ? - ? - ?

A seguir, verifica-se que o gene de mesma posição em *S2* possui valor *6*, e que este valor encontra-se na nona posição de *S*1. Assim, este valor é transportado para a nona posição de *S1'*:

S1' = 3 - ? - ? - 4 - ? - ? - ? - ? - 6

O gene que ocupa a mesma posição em *S2*, possui valor *1*, o qual encontra-se na oitava posição em *S1*. Transporta-se este valor para *S1'*:

S1' = 3 - ? - ? - 4 - ? - ? - ? - 1 - 6

Com a continuação do processo, coloca-se o valor *7* no quinto gene de *S1'*; o valor *2* no segundo gene; o valor *9* no sétimo gene. Mas ao tentar colocar o valor *3* no primeiro gene, observa-se que esta operação já foi realizada.

S1' = 3 - 2 - ? - 4 - 7 - ? - 9 - 1 - 6

Portanto o ciclo está completo, e o passo seguinte é completar os genes de *S1'*, com os respectivos elementos de *S2*.

S1' = 3 - 2 - **5** - 4 - 7 - **8** - 9 - 1 - 6

O segundo filho é obtido através do mesmo processo, começando por *S2*, e resulta em:

S2' = 4 - 9 - **8** - 6 - 2 - **5** - 3 - 7 - 1

Resultados teóricos e empíricos que comparam os operadores PMX, OX e CX são encontrados em Goldberg (1987) e Oliver *et al.* (1987). Eles concluem que o PMX e o OX são semelhantes, e que a maior diferença é que o PMX tende a preservar a posição absoluta do gene (por causa do mapeamento ponto a ponto), enquanto o OX tende a preservar a posição relativa do gene (por causa do preenchimento corrediça ou sequencial dos espaços vazios). Chan e Tansri (1994) analisam os três operadores e discutem empiricamente sobre qual operador é o melhor para o problema do *layout* de fábricas.

■ O Operador LOX

O Cruzamento de Ordem Linear, uma variante do Operador OX, foi elaborado por Faulkenauer (*Apud* Della Croce *et al.*, 1995) para satisfazer a todos os requisitos que o JSS (*Job Shop Scheduling*) exige. Para exemplificá-lo, pode-se considerar dois pontos aleatórios de cruzamento nos cromossomas *S1* e *S2*:

S1 = 3 - 5 - 7 - [6 - 2 - 8]- 9 - 1 - 4

S2 = 8 - 9 - 2 - [4 - 7 - 5]- 3 - 6 - 1

Os valores de *S1* que encontram-se em comum com os valores entre os pontos de cruzamento de *S2* [*4, 7* e *5*], são substituídos por asteriscos:

S1 = 3 - * - * - [6 - 2 - 8]- 9 - 1 - *

Substituir os asteriscos pelos números mais próximos, mantendo a ordem de seqüência, e forçando os asteriscos a ocuparem a região entre os pontos de cruzamento:

S1 = 3 - 6 - 2 - [* - * - *]- 8 - 9 - 1

Transferir o segmento de *S2*, localizado entre os pontos de corte, para a mesma região de *S1*. A operação é então realizada invertendo os papéis, para gerar o descendente *S2'*.

S1 = 3 - 6 - 2 - [4 - 7 - 5]- 8 - 9 - 1

3.10.3. O Operador de Mutação

A Mutação é considerada um mecanismo secundário nas operações dos Algoritmos Genéticos, consistindo nas mudanças aleatórias de um ou mais genes de um cromossoma, e conseqüentemente produzindo albinos [Chan e Tansri,1994]. Albinos são indivíduos com algumas propriedades de cromossomas completamente diferentes da maioria de uma população. Sua função é criar uma política de prevenção que diminua as probabilidades de as soluções serem apanhadas em um ótimo local. Logo, a taxa de mutação é normalmente fixada em um nível muito baixo, tal como nas populações naturais.

A mutação simples (*Simple Mutation* - Goldberg, 1989) consiste da escolha aleatória de uma posição no cromossoma de codificação binária selecionado, e de um sorteio entre os alelos "0" e "1", como exemplificado a seguir, utilizando a terceira posição no *string S1*:

S1 = 1 - 1 - [0] - 1 - 0 - 0 - 1 - 0 - 1

A nova configuração poderia ser:

S1' = 1 - 1 - [1] - 1 - 0 - 0 - 1 - 0 - 1

Segundo Davis (1991), pode-se ainda considerar três tipos de mutação, para problemas em que a ordem de elementos dos cromossomas é uma grandeza importante: a Mutação Baseada na Posição, a Mutação Baseada na Ordem e a Mutação de Mistura.

■ **Mutação Baseada na Posição (*Position-Based Mutation*)**

Duas posições são selecionadas randomicamente, e o conteúdo da Segunda posição é colocado antes da primeira.

S1 = 5 - 1 - [3] - 8 - 5 - 4 - 2 - [6] - 7

A nova configuração é:

S1' = 5 - 1 - [6]- [3] - 8 - 5 - 4 - 2 - 7

■ **Mutação Baseada na Ordem (*Order-Based Mutation*)**

Duas posições são selecionadas randomicamente, e seus conteúdos são intercambiados.

S1 = 5 - 1 - [3] - 8 - 5 - 4 - 2 - [6] - 7

A nova configuração é:

S1' = 5 - 1 - [6] - 8 - 5 - 4 - 2 - [3] - 7

■ **Mutação *Scramble* (*Scramble Mutation*)**

Supondo que a vizinhança das posições é importante, escolhe-se uma sublista randomicamente, e permuta-se a ordem dos conteúdos.

S1 = 5 - 1 - [3 - 8 - 5 - 4] - 2 - 6 - 7

A nova configuração poderia ser:

S1' = 5 - 1 - [8 - 4 - 3 - 5] - 2 - 6 - 7

3.10.4. O Operador de Inversão

As técnicas de cruzamento vistas anteriormente, não levam em consideração a existência de uma ordem entre os genes de um cromossoma. No entanto, em problemas como o da programação de projetos (neste caso, devido às relações de precedência entre as atividades), muito freqüentemente estes operadores genéticos geram descendentes inviáveis.

Assim é que os operadores de reordenação, particularmente o Operador de Inversão, adquirem importância fundamental. Na inversão, uma seção do cromossoma é retirada e reinserida na ordem inversa (vide o exemplo a seguir, onde o segmento entre colchetes de *S1* é invertido). Segundo Reeves [1993], utilizado fora desta situação, o operador de inversão deve possuir a mesma finalidade da mutação e permitir a exploração de regiões do espaço de busca onde o cruzamento não alcançará rapidamente, mas em geral não se pode perder de vista que ele não possui a mesma força recombinativa do cruzamento.

S1 = 1 - 1 - [0 - 1 - 0 - 0] - 1 - 0 - 1

A nova configuração seria:

S1' = 1 - 1 - [0 - 0 - 1 - 0] - 1 - 0 - 1

3.11. Planos de Reprodução

A principal referência na literatura dos Algoritmos Genéticos sobre planos de reprodução constitui a tese de De Jong (1975) [*Apud* Goldberg, 1989]. Sua intenção foi estudar as variações do algoritmo genético a partir de uma versão por ele denominada de Plano Reprodutivo R1 (que Goldberg em 1989, denomina Algoritmo Genético Simples - *Simple Genetic Algorithm*), com as seguintes características:

- Codificação binária do cromossoma;
- Seleção pelo Giro da Roleta, de maneira não proporcional;
- Cruzamento Simples (de um único ponto e acasalamento ao acaso);
- Mutação simples;
- Substituição total de todos os Cromossomas de uma população pelos seus descendentes, a fim de compor a nova população.

De Jong estudou as variações do plano R1, e provou que ele não era um único plano, mas uma família de planos que depende de quatro parâmetros: tamanho da população, probabilidade de cruzamento, probabilidade de mutação e *vazio de geração* (*generation gap* - termo introduzido pelo pesquisador para permitir sobreposição de populações.

Foram investigadas cinco variações do plano reprodutivo R1:

- **Plano Reprodutivo R2 ou Modelo Elitista (*Elitist Model*)**

Introduzido pelo pesquisador, com a finalidade de preservar os melhores cromossomas através da replicação para uma nova população. O Elitismo força os GA's a reterem o mesmo número de melhores indivíduos em cada geração, a fim de que não sejam destruídos pelos operadores de Cruzamento e Mutação, ou perdidos se não forem selecionados para reproduzir. O Elitismo é uma versão artificial da seleção natural enunciada por Darwin, e conforme demonstrado pelo autor, melhora a busca local às custas de uma perspectiva global;

> "Seja a*(t) o melhor indivíduo gerado até o tempo t. Se depois de gerar a população A(t +1), o indivíduo a*(t) não presente, então inclua a*(t) em A(t +1) como sendo o (N+1) membro".

- **Plano Reprodutivo R3 ou Modelo de Valor Esperado (*Expected Value Model*)**

O plano reprodutivo R3 foi criado para reduzir os erros estocásticos da Seleção pelo Giro da Roleta, calculando uma probabilidade de seleção proporcional para a aptidão e selecionando os indivíduos de acordo com a referida distribuição de probabilidade. De Jong demonstrou que o plano R1 é suscetível a duas fontes de erro: (1) as aptidões médias dos *schemata* atuais são calculadas por amostragem finita seqüencial, não é prático calcular através de outra forma; (2) o método de seleção é um processo de alta variância com uma quantidade apropriada de dispersão entre os números de cópias esperado e atual. De Jong tentou reduzir o segundo erro com R3.

- **Plano Reprodutivo R4 ou Modelo do Valor Esperado Elitista (*Elitist Expected-Value Model*)**

Consiste na combinação dos planos R2 e R3. O plano resultou em melhorias consideráveis para funções unimodais e degradação para função com vazios.

- **Plano Reprodutivo R5 ou Modelo do Fator de Aglomeração (*Crowding Factor Model*)**

Difere do plano R1 principalmente porque o processo de substituição dos indivíduos é diferente; neste plano, em lugar da substituição total à cada geração, quando um indivíduo é criado, um outro deve ser imediatamente eliminado.

- **Plano Reprodutivo R6 ou Modelo de Cruzamento generalizado (*Generalized Crossover Model*)**

Neste plano, o *string* é considerado como um anel e um número pré-determinado de pontos de cruzamento CP (*number of crossover points*) é selecionado uniformemente ao redor do círculo. Portanto, existem $C_{CP,L}$ combinações (L: tamanho do *string*) passíveis de serem selecionadas, e à medida em que CP aumenta, cada combinação torna-se menos provável de ser escolhida durante um cruzamento particular. Em outras palavras, segundo as experiências de De Jong, se CP é aumentado, aumentam também as misturas aleatórias de partes dos *strings* como se o cruzamento fosse um simples *embaralhamento*, e consequentemente diminui a preservação dos *squemata* importantes.

3.12. Decisões para Implementar um Algoritmo Genético

O sucesso de um AG utilizado em um problema que contenha as características descritas no item 3.3, depende principalmente de quatro decisões descritas a seguir em forma de perguntas:

- **Qual Codificação Usar?**

Esta é a decisão fundamental do sucesso dos Algoritmos Genéticos, e segundo Mitchell (1996), ainda não há nenhuma conclusão rigorosa sobre qual codificação conduzirá ao melhor resultado para um problema a ser solucionado. Para muitas aplicações, tem sido mais natural utilizar alfabetos de poucos caracteres ou números reais do que a codificação binária;

- **Como Realizar a Seleção?**

A Seleção tem que ser equilibrada com o cruzamento e a mutação em relação à diversidade: meios de seleção muito fortes reduzirão a diversidade necessária para trocas genéticas posteriores e progresso da população; seleção muito fraca resultará em evolução muito lenta;

- **Quais Operadores Usar (Além da Seleção)?**

Esta decisão depende do problema abordado, bem como da estratégia de codificação do cromossoma e do tipo de relacionamento que cada gene guarda com os demais;

- **Como ajustar os Parâmetros?**

Ainda não existem resultados conclusivos sobre o estabelecimento de parâmetros como o tamanho da população, a taxa de cruzamento e a taxa de mutação.

Um fato interessante é que os parâmetros interagem de forma não linear uns com os outros, de modo que não podem ser otimizados separadamente. Os experimentos de De Jong [*Apud* Reeves, 1993], indicam o melhor tamanho de população entre 50-100 indivíduos, taxa de cruzamento em aproximadamente 0.6 e taxa de mutação em aproximadamente 0.001; Grefenstette (1986) obteve tamanho de população 30, taxa de cruzamento 0.95 e taxa de mutação 0.01; Schaffer, Caruana, Eshelman e Das (1989) chegaram aos seguintes valores respectivamente: 20-30, 0.75-0.95 e 0.005-0.01.

3.13. A Estrutura Básica do Algoritmo

A estrutura básica do algoritmo, expressa através da representação em pseudocódigo, pode ser vista na figura 3.2.

```
Algoritmo Genético

Inicialização
    Gere a População Inicial {de n Indivíduos};
    Avalie o Fitness de cada Indivíduo;
Enquanto a Condição de Parada não for Verdadeira
    Repita até que a nova População esteja Completa
        Selecione dois Pais da Geração;
        Aplique o Cruzamento para gerar dois Descendentes;
        Aplique Mutação nos Descendentes;
        Avalie o Fitness dos Descendentes;
        Coloque os Descendentes na nova População;
    Geração Atual := Nova População;
```

Fig. 3.2 - Estrutura Básica de um Algoritmo Genético, em Pseudo-Código. Fonte: Goldberg, 1989.

3.14. Conclusão

Um Algoritmo Genético trabalha com base na geração e na evolução contínua de populações de cromossomas. Cada estrutura de cromossoma codifica uma solução do problema e representa um ponto no espaço de busca; portanto, sua aptidão está relacionada ao valor da função objetivo.

Os AG's constituem um instrumento novo de pesquisa, capaz de ser utilizado em diversas áreas das atividades humanas. Sua principal vantagem reside na diversidade genética: a cada passo do processo de busca, um conjunto de candidatos é considerado e envolvido na criação de novos candidatos. Existem outras importantes vantagens, mas apesar disto, a utilização desta técnica heurística deve estar sempre restrita aos problemas cujas características sejam coerentes com seu campo de

aplicação. Em termos de comparação com outros métodos heurísticos, pode-se frisar que apesar dos muitos estudos e experimentações feitos, estes ainda são insuficientes para estabelecer conclusões precisas e rigorosas [Reeves, 1993].

Um bom resultado a ser obtido pelos Algoritmos Genéticos, dependerá do rendimento de detalhes como o método de codificação das soluções candidatas, os tipos de operadores genéticos utilizados, os ajustes de parâmetros e outros critérios particulares. Um AG na sua forma básica pode ser insuficiente para um determinado tipo de problema, por exemplo: a codificação *bit-string* pode ser inadequada; a seleção através da proporcionalidade do *fitness* pode conduzir à convergência prematura; os operadores de cruzamento e de mutação podem tornar inviáveis as soluções descendentes, bem como podem destruir bons esquemas dos cromossomas atuais e um ajuste inadequado dos parâmetros pode diminuir a performance global. Por este motivo, as decisões sobre cada passo da elaboração do algoritmo devem ser analisadas cuidadosamente, visto que existem muitas possibilidades de implementação e o relacionamento entre os objetos de decisão geralmente não estão suficientemente claros e simples.

Finalmente, vale salientar que embora alguns elementos sejam gerados fortuitamente, a pesquisa dos Algoritmos Genéticos não é aleatória; ela procura explorar eficientemente a informação histórica. Segundo Reeves (1993), da perspectiva da Pesquisa Operacional, a idéia de um AG pode ser entendida como o uso inteligente de uma busca aleatória.

CAPÍTULO IV

O MÉTODO DE SOLUÇÃO PROPOSTO

4.1. Introdução

Este capítulo é o cerne do presente trabalho de tese. Inicialmente apresenta-se a definição do problema, formalizando as idéias expostas no primeiro capítulo. A seguir desdobra-se o método de solução, fundamentado nos Algoritmos Genéticos e nas seguintes heurísticas específicas: grafo atividade-no-nó, Método da Linha de Balanço e Heurísticas Baseadas em Regras de Prioridade..

4.2. Definição do Problema

Considera-se um projeto de construção de um edifício dotado de múltiplos pavimentos-tipo. As premissas que descrevem o problema geral, são constituídas pelas suposições do RCPSP clássico (mostrado no capítulo II), com as devidas modificações e acréscimos:

- Um projeto consiste de diferentes atividades, as quais são representadas no formato atividade-no-nó (*activity-on-the-node format* - um grafo orientado e acíclico, no qual os nós representam as atividades e os arcos representam as restrições de precedência). Duas atividades fantasmas são introduzidas: a atividade *1* representa a atividade de início do projeto e é a predecessora direta ou indireta de toda atividade do projeto, enquanto a atividade *n* denota a atividade final do projeto e é sucessora direta ou indireta de todas as atividades;

- A estrutura de precedências das atividades que compõem o projeto é perfeitamente conhecida e não pode sofrer modificações durante o processamento;

- As atividades estão relacionadas através de relações de precedência final-início, com um atraso de tempo zero, implicando que nenhuma atividade pode ser iniciada antes que todas as suas predecessoras tenham sido completadas. No caso das atividades repetitivas, este sincronismo é obtido através da análise de pavimento a pavimento;

- Todas as informações que alimentam o projeto, são de expressão determinística;

- As atividades podem ser de dois tipos: (1) as repetitivas, quando são realizadas nos pavimentos-tipo da torre; e (2) as não-repetitivas, quando são realizadas em quaisquer outros pavimentos abaixo ou acima da torre;

- A taxa de progresso das atividades repetitivas é considerada constante;

- Toda atividade repetitiva inicia no primeiro pavimento-tipo e finaliza no último pavimento-tipo, ou vice-versa;

- A quantidade de trabalho de uma atividade repetitiva não sofre variações ao longo dos pavimentos.

- Nenhuma atividade pode ser interrompida depois de iniciada (não é permitida a preempção de atividade);

- Nenhuma data de início ou de término é imposta a qualquer atividade do projeto (excluindo as atividades inicial e final, que são *dummies*);

- Cada atividade pode possuir um ou mais modos de execução, sendo que cada modo refere-se ao incremento de uma equipe de trabalho;

- Existem restrições espaciais que limitam o número de equipes de trabalho para cada atividade;

- Existem restrições de recurso monetário à cada período de tempo, decorrentes da Curva de Agregação planejada para o projeto;

- Os tempos de preparação são considerados desprezíveis ou incluídos nas durações;

- As despesas ligadas à cada atividade, são de natureza direta e são constantes durante todo o processamento;

- Cada entrada de recurso monetário disponível define o final de um período de tempo; o prazo do projeto corresponde à unidade de tempo da última entrada deste tipo de recurso;

Os objetivos da programação são:

- Minimizar as diferenças entre os valores da curva de agregação de recursos planejada e os valores das despesas diretas do projeto, por período de tempo; bem como promover a continuidade do trabalho nas atividades repetitivas.

Segundo a classificação de Belmann *et al.* (1982), trata-se de um Problema de Programação da Produção, pertencente à classe dos Problemas de Programação de Projetos, e de acordo com a classificação de Davis (1973), pode-se subclassificá-lo como um Problema de Programação de Projetos com restrição de Recursos (*RCPSP: Resource-Constrained Project Scheduling Problem*). Ele enfoca a questão de como um conjunto de atividades inter-relacionadas tecnologicamente, sob

restrição de recursos, pode ser programado de forma a atingir um objetivo da administração.

O problema refere-se a um único projeto (*single-project scheduling*), de natureza determinística, no qual as atividades são interligadas pelo tipo Final-Início, sem possibilidade de interrupção, com múltiplos modos (*multi-mode*) de execução, enfatizando dois tipos de recursos: o monetário, o qual é do tipo duplamente restrito, e o recurso mão-de-obra, o qual é do tipo renovável. Portanto, caracterizando o problema de múltiplos recursos (*multiple resources*). Face à existência de um objetivo de programação, pode-se caracterizá-lo como sendo de objetivo único (*single objective*); a continuidade do trabalho nas atividades repetitivas pode ser obtida através do uso de um Método de Programação Linear.

4.3. O Método de Solução Proposto

O método de solução proposto baseia-se na abstração de que o projeto possui quatro atributos: (1) um objetivo a ser alcançado, o qual é personificado por uma função objetivo; (2) um conjunto de atividades finitas à serem executadas; (3) um conjunto de recursos necessários à sua implementação e (4) um conjunto de restrições à livre programação. O método deve gerar uma solução ou resultado da função objetivo, que satisfaça às restrições e que seja o melhor possível, no que se refere aos objetivos citados na definição do problema. Nesta função, os valores das variáveis dependentes são extraídos de uma circunstância de programação das atividades.

Na terminologia adotada, denomina-se *programa* à toda lista tripla que define um seqüenciamento ou ordem de programação, bem como os modos de produção adotados, e as datas de início, para as atividades. Denomina-se *programa viável* àquele que satisfaz às restrições, e em extensão, *programa inviável* àquele que não satisfaz à pelo menos uma das restrições. Denomina-se *melhor programa* àquele que satisfazendo às restrições, resulta no melhor valor da função objetivo.

O mecanismo de funcionamento do método proposto, baseia-se no refinamento contínuo de uma população inicial de programas viáveis, através de um processo evolutivo concebido com base nos Algoritmos Genéticos. Na primeira iteração, o algoritmo genético avalia cada indivíduo (programa) da população atual, e posteriormente gera uma nova população utilizando a reprodução com Elitismo, em conjunto com os operadores de Seleção, Cruzamento, Mutação e Inversão. Os programas gerados por Elitismo, são transferidos integralmente para a nova população, enquanto que os programas gerados pelos outros operadores, transferem apenas os atributos de ordem e modo. Para definir as novas datas de início e de término dos programas incompletos, é necessário satisfazer às restrições do problema. Feito este passo, o algoritmo genético reinicia outra iteração. Este processo continua até que o número de iterações convencionado seja atingido, preservando os melhores programas.

As soluções iniciais são geradas através do Esquema de Programação Serial (vide cap. II). Na lista ordenada, a primeira atividade é sempre a *dummy* inicial; a

atividade seguinte é uma sucessora, a qual é escolhida randomicamente se estiver em paralelo com outras; este processo continua até que a última atividade (*dummy* final) esteja colocada na lista. O modo de produção de cada atividade, também é escolhido aleatoriamente dentro dos limites estabelecidos. As datas de início são determinadas de maneira a satisfazer às restrições de lógica horizontal, às restrições de *buffer* imposto, às restrições de lógica vertical, às restrições de mão-de-obra e às restrições de recurso monetário, exatamente nesta seqüência. No caso das atividades repetitivas, o cumprimento das restrições é realizado pavimento a pavimento, através do uso do Método da Linha de Balanço. Este modelamento pode ser visto com mais detalhes no item 4.4. Por meio de arranjos de ordem e duração, é possível gerar um grande número de programas distintos.

Na estruturação concebida, o Esquema de Programação Serial e os processadores de restrições estão contidos no denominado Módulo de Programação (item 4.5.1), enquanto o algoritmo genético está contido no denominado Módulo de Busca (item 4.5.2).

4.4. O Modelamento do Projeto

De acordo com a definição do problema, a estruturação lógica do projeto no modelo proposto é obtida através da analogia com um grafo orientado, onde os nós representam as atividades e os arcos representam as interrelações entre estas atividades. Este tipo de representação constitui um sistema discreto, já que as mudanças de estado ocorrem em instantes precisos, em oposição a um sistema contínuo.

A representação indicial unidimensional (devido as atividades serem representadas pelos nós do grafo, e não pelos arcos) para todas as variáveis e a ausência de atividades fictícias (exceto no início e no final da rede) são duas vantagens imediatas da representação atividade-no-nó; no entanto, estas qualidades tornam-se menos relevantes quando o projeto refere-se à *construção de massa* [O'Brien, 1969]. Suhail e Neale (1994) explicam que esta representação não é orientada para prover a continuidade das equipes nas atividades repetitivas, o que é um requisito fundamental para uma construção de caráter linear. Além disto, Rahbar e Rowing (1992) acusam ainda sua incapacidade de distinguir as taxas de progresso das atividades, assim como o problema de o número de pavimentos a ser completado em algum período de tempo não ser claramente visível.

Ao implementar um modelo baseado em grafos, percebe-se dois tipos de lógica: (1) a interligação lógica absoluta e (2) a interligação designada pela seqüência sugerida, em oposição à seqüência pura. Como exemplo, na figura 4.1 visualiza-se a estruturação das atividades repetitivas de um edifício com três pavimentos-tipo, onde cada pavimento é representado por uma denominada rede básica, que consiste das atividades pertencentes àquele pavimento: as interligações inerentes à rede básica constituem o primeiro tipo de lógica, enquanto as interligações entre pavimentos constituem o segundo tipo. Assim, a decisão de priorizar uma atividade na programação, pode incorrer em dois tipos de erros [Suhail, 1993]: (1) o progresso fora de seqüência

(out-of-sequence progress), que diz respeito à transgressão da lógica absoluta, e (2) o progresso fora de lógica (*out-of-logic progress*), que diz respeito à transgressão da sucessão de trabalho entre as unidades repetitivas.

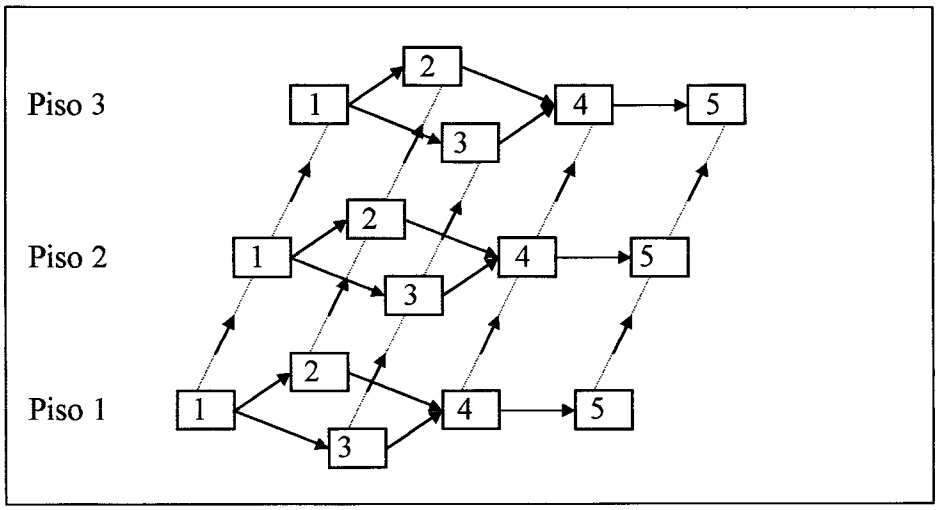


Fig. 4.1 - Tipos de lógica em um modelo de grafos: (1) a interligação lógica absoluta, representada pelos arcos do grafo, e (2) a interligação designada pela seqüência sugerida, representada pelas setas tracejadas. Fonte: do autor.

A representação das atividades repetitivas utilizando somente uma estrutura de grafos, conduz a um processo de programação tedioso, cujas estratégias dependem das decisões da administração e não das dependências exigidas pelas atividades, visto que o modelo prevê apenas a interligação lógica absoluta [Carr e Meyer, 1974; Pilcher, 1976 e Clough e Sears, 1991 e Rahbar e Rowind, 1992]. Para suprir esta deficiência, vários modelos de integração entre os métodos baseados em grafos e os métodos de programação de projetos lineares foram apresentados em trabalhos científicos, tal como em Schoderbeck e Digman, 1967; Carr e Meyer, 1974; Birrel, 1980; Perera, 1982; Al Sarraj, 1990; Russel e Wang, 1993 e Suhail e Neale, 1994].

Seguindo este caminho, este trabalho busca integrar os dois tipos de métodos, adotando um grafo para modelar o projeto como um todo, e um método de programação de projetos lineares para modelar cada atividade repetitiva e assim garantir de forma mais eficiente, as sincronizações término-início em cada pavimento. Portanto, a interligação lógica absoluta fica garantida pelo grafo, enquanto a sincronização e a continuidade (interligação designada pela seqüência sugerida) das equipes de trabalho nas atividades repetitivas ficam garantidas pelo outro método. A figura 4.2 , composta de seis atividades não fictícias, das quais as atividades *3*, *5* e *6* são repetitivas, evidencia esta integração.

Considerando este tipo de integração, o modelo proposto passa a ser classificado de acordo com o grau de abstração desejado ou do ponto de vista do

observador: cada uma das linhas de balanço constitui um sistema contínuo em si, mas o projeto como um todo, consiste em um sistema discreto. Um outro importante aspecto da adoção de um método de programação voltado para projetos lineares, é a capacidade de orientar a programação das atividades repetitivas pelos recursos e não pelas atividades, o que segundo Trimble (1984), é mais realista.

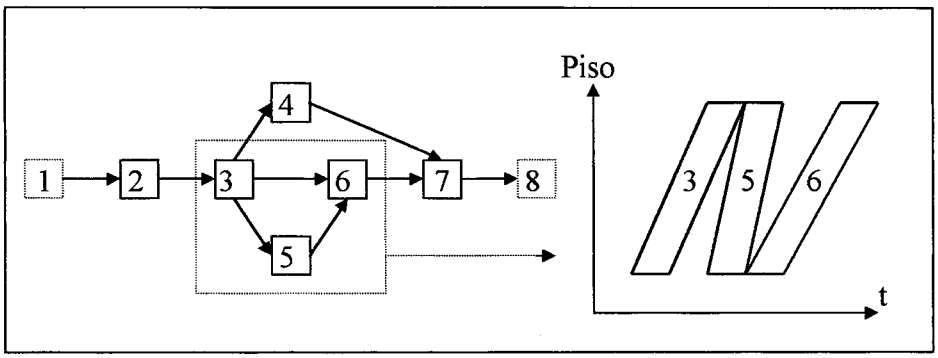


Fig. 4.2 - Integração entre o grafo representativo do projeto e o método de programação linear representativo das atividades repetitivas. Fonte: do autor.

As atividades repetitivas são executadas no edifício em duas direções, uma horizontal e uma vertical [Thabet e Beliveau, 1994]. O resultado deste fato é que surgem restrições de lógica horizontal e restrições de lógica vertical, a serem satisfeitas pela programação. Enquanto o primeiro tipo de restrição controla a sucessão lógica entre atividades em um mesmo piso, o segundo tipo controla as relações de dependência entre atividades executadas em diferentes pisos. Thabet e Beliveau (1994), descrevem um procedimento estruturado para incorporar ambos os tipos na programação de edifícios, denominado HVLS (*Horizontal and Vertical Logic Scheduling*), cujos princípios e síntese dos possíveis casos de programação, são também utilizados no modelo proposto neste trabalho de tese. Outras importantes contribuições neste sentido encontram-se em Thabet (1992) e Thabet e Beliveau (1993), que associaram o HVLS com outros métodos de programação para desenvolver o SCaRC (*Space Constrained and Resource Constrained*), um sistema especialista para a programação de pavimentos repetitivos em edifícios.

4.4.1. Validação do Grafo Atividade-No-Nó

Para ser capaz de representar o projeto, o grafo suporte possui algumas regras de validação lógica e computacional: (1) a unicidade dos nós inicial e final, de modo a existir somente um ponto de início e outro de término para o processamento dos algoritmos que envolvem o sequenciamento das atividades; e (2) a inexistência de ciclos ou circuitos, que conduzem a implementação dos algoritmos à um *looping* interminável. Na figura 4.3 pode-se visualizar exemplos de transgressões a cada uma destas regras.

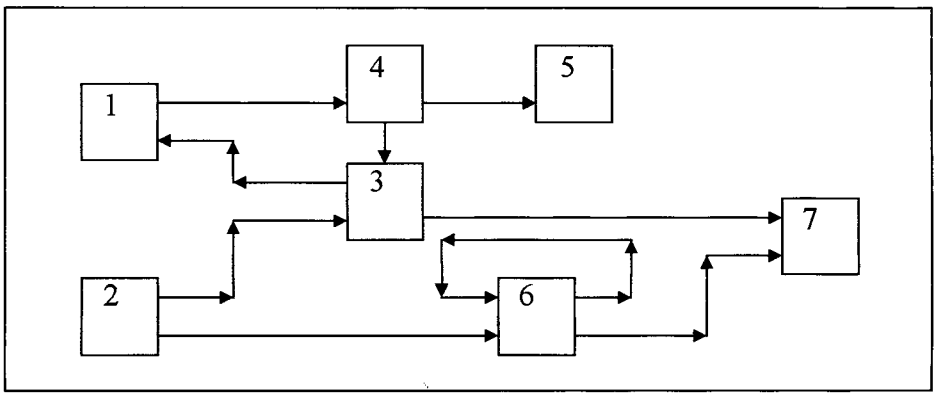


Fig. 4.3 - Erros em uma grafo atividade-no-nó: (a) a presença de mais de uma atividade de início [1 e 2] ou de final [5 e 7]; (b) a presença de ciclos na rede [6]; e (c) a presença de circuitos na rede [1, 4, 3, 1]. Fonte: do autor.

4.4.2. Tipo de Ligação Previsto entre Atividades

É admitida somente a ligação denominada final-início, segundo a qual a atividade sucessora só pode ser iniciada em uma unidade de tempo maior que a data de término da atividade antecessora. A regra é válida também para as atividades repetitivas, considerando-a pavimento a pavimento.

4.4.3. A Numeração dos Pavimentos

Os pavimentos não repetitivos, que no modelo concebido do edifício estão situados somente abaixo e acima da torre, não necessitam de numeração porque as atividades neles realizadas, são modeladas somente pelo grafo suporte. Os pavimentos repetitivos, por exigência do Método da Linha de Balanço, são numerados no sentido ascendente (do mais baixo para o mais alto) sem levar em consideração a localização relativa no edifício.

4.4.4. O Método de Programação de Projetos Lineares Adotado

O Método da Linha de Balanço é o método de programação de projetos lineares adotado neste trabalho. Sua utilização por si só, garante manter a continuidade das equipes de trabalho, satisfazendo à um dos objetivos do problema a ser solucionado. Outros métodos citados no capítulo II, também poderiam ser usados sem prejuízo.

No exemplo da figura 4.4, apresenta-se um pequeno projeto composto por três atividades repetitivas *a*, *b* e *c*, dispostas seqüencialmente em um grafo. À sua direita, o mesmo projeto está representado por uma linha de balanço, evidenciando os quatro pavimentos em que as atividades repetem-se continuamente. Cada uma das barras

horizontais de uma mesma atividade, representa sua realização em um determinado pavimento, em escala de tempo. O projeto inicia-se no tempo t_i, que corresponde também ao início dos trabalhos no primeiro pavimento; o término do primeiro pavimento acontece no tempo t_{pl} e o término do projeto acontece no tempo t_f. Pode-se deduzir que cada equipe que trabalha em uma determinada atividade, desloca-se para o pavimento seguinte após o término da execução no pavimento atual.

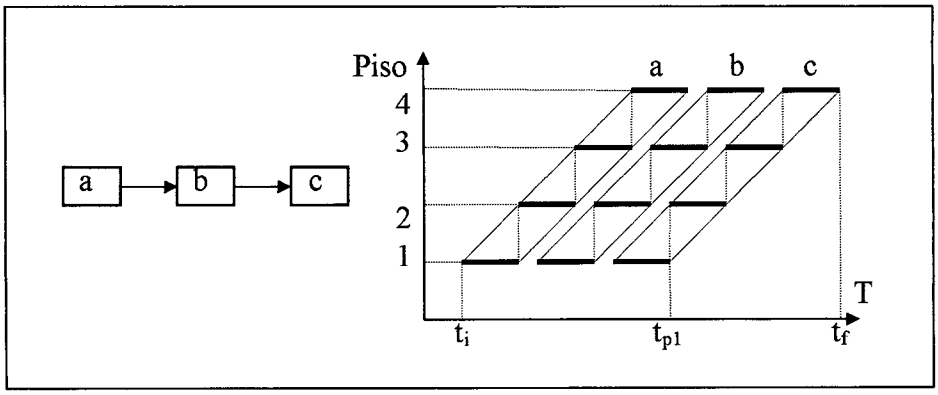


Fig. 4.4 - Programação de atividades repetitivas: dado um projeto que envolve três atividades consecutivas, a serem realizadas em quatro unidades repetitivas. Sua representação através de um grafo atividade-no-nó e através do Método da Linha de Balanço, podem ser vistos à esquerda e à direita da figura, respectivamente. Fonte: do autor.

4.4.5. Tipo de Programação Adotado para o Método da Linha de Balanço

Existem dois métodos de programação por Linha de Balanço: o Método da Programação Paralela, e o Método da Programação de Recursos. A programação paralela está ligado ao princípio do ritmo uniforme de produção. Assim, todas as atividades repetitivas do projeto devem possuir a mesma inclinação (ritmo) em relação ao eixo horizontal. Na Programação de Recursos, as atividades repetitivas são programadas em função de seus ritmos naturais, resultando em programas mais longos que os obtidos com o emprego da programação paralela.

Ao comparar os dois métodos, percebe-se que o tempo mínimo de realização para um projeto, é obtido quando se utiliza a Programação Paralela. Além disto, neste método há uma grande redução dos tempos vazios, mas existe a desvantagem do aumento de custo e do ritmo artificial das atividades. Portanto, a Programação Paralela apresenta maior dificuldade de planejamento, mas resulta em programas com menor ociosidade. Em projetos do mundo real, especialmente os de construção civil, é difícil e às vezes impossível conseguir a uniformização dos ritmos de produção, visto que a maioria das atividades são distintas entre si relativamente ao produto a ser construído, às quantidades de trabalho, bem como à formação, produtividade e número de equipes a ser alocado; por este motivo, neste trabalho adota-se a Programação de Recursos. Na figura 4.5, mostra-se os dois tipos de programação.

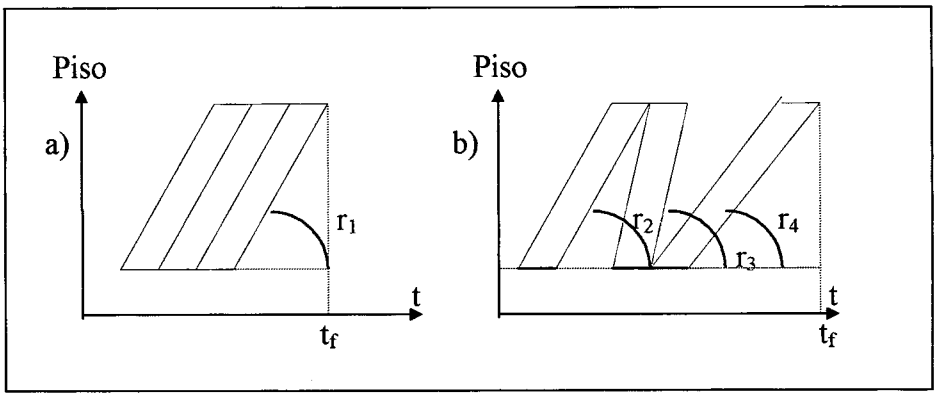


Fig. 4.5 - Tipos de programação pelo Método da Linha de Balanço: (a) a programação paralela e (b) a programação por recursos. Em ambos os casos, verifica-se o ritmo de produção r_i, uniforme somente para a programação paralela. Fonte: do autor.

4.4.6. Combinações Possíveis das Atividades Repetitivas em Relação ao Sentido de Execução

Na construção de um edifício, cada atividade repetitiva possui um sentido de execução na direção vertical. A maioria das atividades é implementada no sentido ascendente, mas algumas atividades fogem a esta regra: umas por razões de segurança, outras para prevenir danos nos trabalhos prontos, ou outras razões. Os serviços de revestimento e pintura externos e o serviço de limpeza final constituem atividades executadas no sentido descendente. Em relação ao sentido de execução, existem quatro combinações duas a duas destas atividades, como visto na figura 4.6.

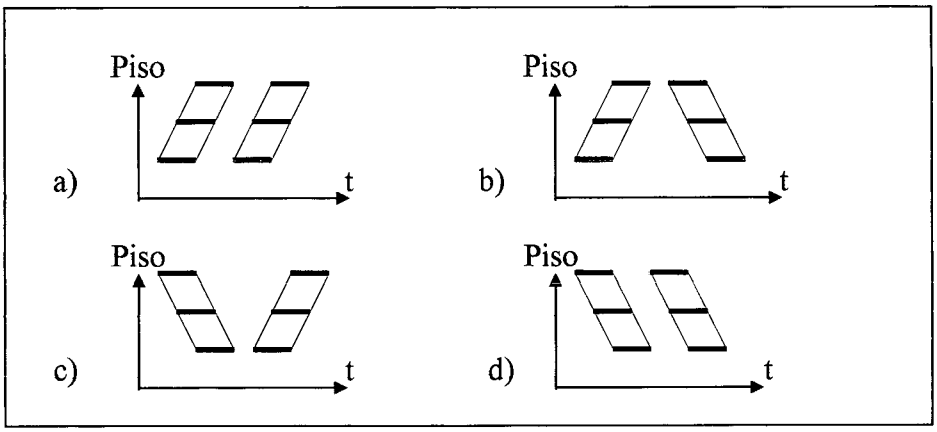


Fig. 4.6 - Combinações Possíveis entre os dois sentidos das atividades repetitivas, segundo a direção vertical: (a) duas atividades ascendentes; (b) uma atividade ascendente e outra descendente; (c) uma atividade descendente e outra ascendente; (d) duas atividades descendentes. Fonte: do autor.

4.4.7. Quantidade de Trabalho por Pavimento

Vorster e Bafna (1992) sugerem que os projetos lineares podem ser divididos em duas categorias. Na primeira, encontram-se os Projetos Repetitivos Típicos,

cujas atividades possuem a mesma duração para todas as unidades [Lumsden, 1968; Carr e Meyer, 1974; Arditi e Albulak, 1986; Al Sarraj, 1990] e na segunda, encontram-se os Projetos Repetitivos Não Típicos, cujas durações variam por unidade (exemplo: o tempo de escavação das fundações, pode variar de um ponto para outro) [Selinger,1980; Johnston, 1981; Chrznowski e Johnston, 1986; Russel e Caseldon, 1988 e Mosheli, 1993]. No método proposto, supõe-se que os pavimentos sejam perfeitamente típicos ou que a variação seja desprezível em termos de programação de obras. Na figura 4.7, o caso *a* é aceito, mas não o caso *b*.

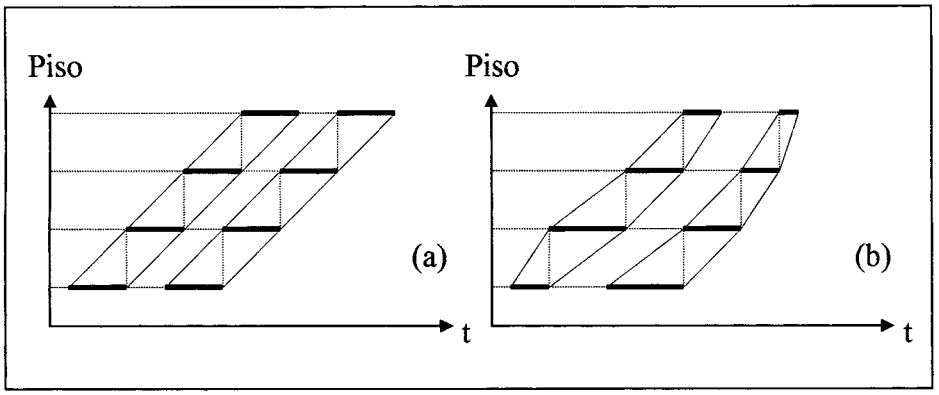


Fig. 4.7 - Exemplo de atividades de (a) Projeto Repetitivo Típico e de (b) Projeto Repetitivo Não-Típico, representadas pelo Método da Linha de Balanço. Fonte: do autor.

4.4.8. Número de Equipes por Pavimento Repetitivo

O método proposto adota a premissa de que o número de equipes não pode variar de um pavimento para o outro, em uma mesma atividade repetitiva; dito de outra forma, a taxa de progresso de uma atividade ou o seu ritmo de produção é considerado constante. Uma conseqüência importante desta decisão, é que as atividades repetitivas não pode sofrer realocação de mão-de-obra após as datas de início. Na figura 4.8, considerando-se que a quantidade de trabalho é constante por pavimento, o caso *a* é aceito, mas não o caso *b*.

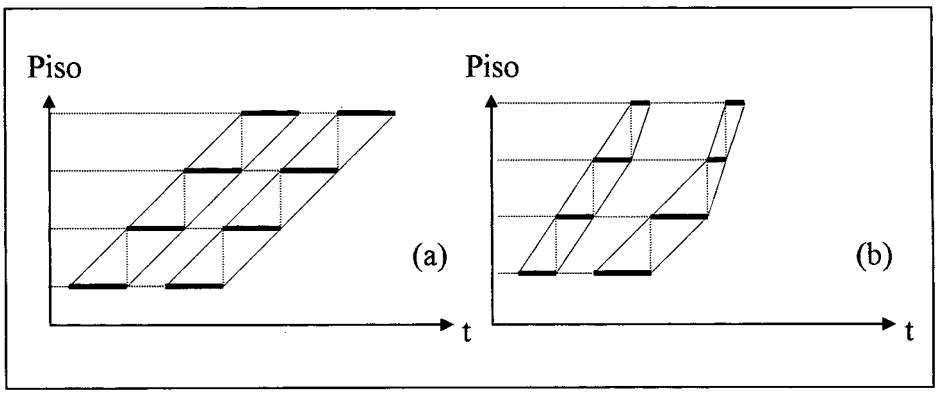


Fig. 4.8 - Exemplo de atividades com (a) número constante de equipes por pavimento e (b) variação do número de equipes nos pavimentos. Fonte: do autor.

4.4.9. O Modo de Produção das Atividades

Toda atividade *j* possui um ou mais modos *m* de produção, o qual possui como atributos o número de equipes $NEq_{m,j}$, a duração total $d_{m,j}$ medida em unidades de tempo ou dias, e uma despesa diária $DespDiária_{m,j}$ (vide figura 4.9). No caso das atividades repetitivas, cada modo possui mais dois atributos: o ritmo de produção $R_{m,j}$ e a duração em cada pavimento-tipo $dTipo_{m,j}$.

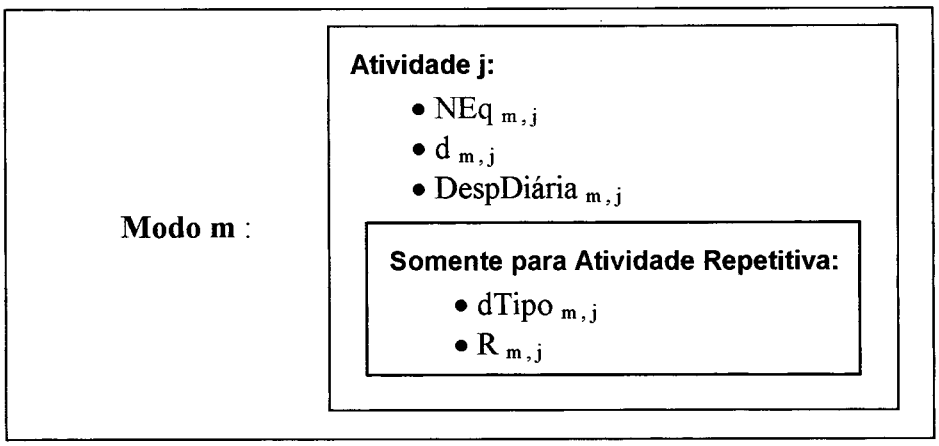


Fig. 4.9 - Esquematização do Modo de Produção de uma Atividade. Fonte: do autor.

O número de modos de produção de uma atividade depende da formação da equipe de trabalho e do número de equipes passível de ser alocado. O método proposto funciona melhor com equipes formadas com o menor número de oficiais possível, porque resultam em incrementos pequenos de duração e despesa, facilitando a redução entre os desvios de prazo e de recurso monetário no processo de programação. Em outras palavras, a idéia é a de que os modos sejam pontos discretos próximos uns dos outros. À rigor, o menor número de equipes é a unidade, mas para reduzir o espaço de busca do problema, adota-se a medida de eliminar os valores considerados incoerentes pelo bom senso do construtor (Como exemplo, pode-se eliminar o modo que aloca somente uma equipe formada por um oficial e ajudantes, para o levantamento da alvenaria de um grande pavimento). O número máximo de equipes é discutido no item 4.5.1.3.

Algumas maneiras para calcular o valor das variáveis associadas ao modo de produção são mostradas a seguir, embora seja recomendável que o programador de projetos utilize o procedimento que lhe seja mais apropriado e acessível.

■ A Determinação das Durações

Seja a operação *Round [x ; 0,2]* com $x \in \Re$, a qual resulta no menor número inteiro mais próximo do valor de x se $x \leq 0,2$ e resulta no maior número inteiro mais próximo em caso contrário. A duração $d_{m,j}$ de uma atividade *j* com modo *m* pode ser obtida por uma das expressões vistas a seguir.

$$d_{m,j} = Round\,[\,(Q * P) / (NOfEq_j * NEq_{m,j} * Jdiária)\,;\,0{,}2];\qquad [1]$$

Com Q = Quantidade de trabalho;
P (homens-hora / Q) = Índice de produtividade;
NOfEq (homens) = Número de oficiais na equipe;
$NEq_{m,j}$ = Número de Equipes da atividade *j* no modo *m*;
Jdiária (horas / dia) = Jornada diária de trabalho;

Quando se dispõe de índices de produtividade expressos em *Q/dia*, pode-se utilizar a seguinte expressão:

$$d_{m,j} = Round\,[\,Q / (P * NEq_{m,j})\,;\,0{,}2];\qquad [2]$$

Na ausência de índices de produtividade, o valor numérico de $d_{m,j}$ pode ser obtido por estimativa resultante de consenso do corpo técnico envolvido no projeto:

$$d_{m,j} = Round\,[\,\text{estimativa técnica}\,;\,0{,}2\,];\qquad [3]$$

Uma observação importante é que a quantidade de trabalho *Q* das atividades repetitivas diz respeito à somatória do valor desta variável em todos os pisos em que ela ocorre. Logo, a duração $d_{m,j}$ refere-se à faixa de tempo compreendida entre o início da atividade *j* no primeiro pavimento-tipo e a data de término desta atividade no último pavimento-tipo.

A duração $dTipo_{m,j}$ de uma atividade repetitiva pode ser obtida através da seguinte expressão:

$$dTipo_{m,j} = d_{m,j} / Ntipo;\qquad [4]$$

Onde NTipo = Número de Pavimentos-tipo;

Um procedimento alternativo para calcular o valor de $dTipo_{m,j}$, consiste em utilizar uma das expressões de $d_{m,j}$ ([1], [2] ou [3]), bastando para isto usar a quantidade de trabalho de apenas um pavimento-tipo. Neste caso, a operação *Round* não seria realizada, por exemplo através da expressão [1]:

$$dTipo_{m,j} = (Q * P) / (NOfEq_j * NEq_{m,j} * Jdiária);\qquad [5]$$

Com Q = Quantidade de trabalho em um pavimento-tipo;

E o valor de $d_{m,j}$ seria obtido por:

$$d_{m,j} = Round\,[\, dTipo_{m,j} * NTipo \,] \qquad [6]$$

Ao utilizar este procedimento alternativo, o valor de $dTipo_{m,j}$ deve ser corrigido utilizando a expressão [4], devido ao arredondamento da duração $d_{m,j}$.

A figura 4.10 focaliza o processo da mudança de modo em uma atividade repetitiva *j* dotada de três modos de execução. Quando somente uma equipe é designada, obtém-se a duração mais longa $d_{1,j}$; ao utilizar duas equipes a duração diminui para $d_{2,j}$; e ao utilizar três equipes sua duração se reduz a $d_{3,j}$. Analogamente, a duração mais longa em um pavimento-tipo é representada por $dTipo_{1,j}$; a duração para duas equipes é representada por $dTipo_{2,j}$; e a menor duração diz respeito à $dTipo_{3,j}$.

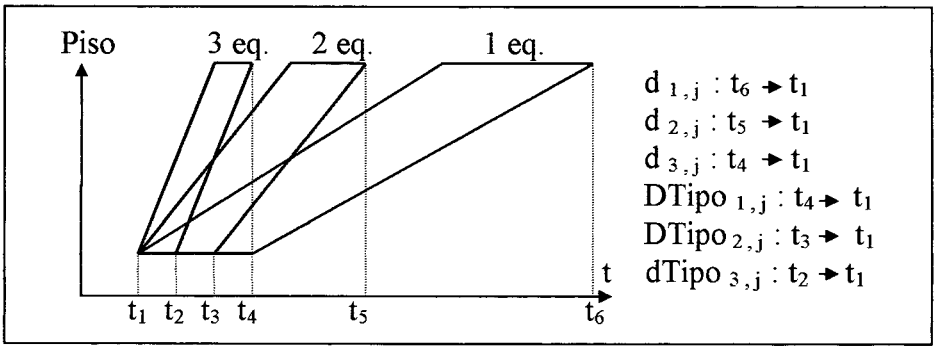


Fig 4.10 - Ilustração das durações das atividades de acordo com os modos de produção. A atividade *j* possui três modos de produção e três durações de cada tipo $d_{NEq,j}$ e $dTipo_{NEq,j}$. Fonte: do autor.

A duração de uma atividade repetitiva é máxima quando somente uma equipe a executa continuamente do primeiro ao último pavimento-tipo. Através do incremento de equipes surgem durações menores até que aconteça uma das três situações: (1) o número máximo de equipes em um mesmo pavimento seja atingido (vide item 4.5.1.3); (2) o incremento de uma equipe cause uma redução inferior a um dia na duração (o modelo não considera frações de dias neste caso); ou (3) o incremento de uma equipe resulte em uma duração menor que um dia.

■ A Despesa Diária

A despesa diária $DespDiária_{m,j}$ de uma atividade *j* sob modo *m*, pode ser obtida através da expressão seguinte:

$$DespDiária_{m,j} = DespTotal_j / d_{m,j} \qquad [7]$$

Onde $DespTotal_j$ = Despesa Total da atividade *j*;

■ **O Ritmo das Atividades Repetitivas**

O ritmo de produção $R_{m,j}$ da atividade *j* com um dado modo *m*, pode ser calculado pela expressão mostrada a seguir:

$$R_{m,j} = 1 / dTipo_{m,j} \quad [8]$$

■ **Um Exemplo de Aplicação:**

O exemplo de aplicação das expressões vistas neste presente item possui os seguintes dados: considerando-se uma jornada de trabalho de 44 horas/semana e considerando-se 5 dias úteis por semana, obtém-se 8,8 horas/dia de trabalho. Inicialmente, para calcular a duração da atividade não repetitiva *Concretagem*, de um volume de 80 m^3, utilizando três equipes 2P+2S (2 pedreiros e dois serventes), pode-se recorrer à tabelas do tipo TCPO (Tabela de Composição de Preços para Orçamentos), de onde retira-se taxa de produtividade 1,60 hh/m3. A maior duração acontece com um número de equipes $NEq_{mj} = 1$, e utilizando a expressão [1] se obtém:

$$d_{m,j} = Round\,[80 * 1,6 / (2 * 1 * 8,8)] \therefore d_{m\,j} = 8 \text{ dias}$$

A duração da atividade *j* para $NEq_{mj} = 3$, utilizando a expressão [1] é obtida através do seguinte cálculo:

$$d_{m,j} = Round\,[80 * 1,6 / (2 * 3 * 8,8)] \therefore d_{m,j} = 3 \text{ dias}$$

Se a atividade *Concretagem* fosse executada em cada piso de uma torre de 20 pavimentos-tipo, mantendo os demais itens, a duração total usando [7] seria:

$$d_{m,j} = 2,4242 * 20 = 48,48 \text{ dias} \therefore d_{m,j} = 49 \text{ dias}$$

E o valor corrigido de $dTipo_{m,j}$ utilizando [4], seria:

$$dTipo_{m,j} = 49 / 20 \therefore dTipo_{m,j} = 2,45 \text{ dias}$$

O ritmo para esta atividade repetitiva, no contexto de $d_{m,j}$ e $dTipo_{m,j}$ seria:

$$R_{m,j} = 1 / 2,45 \therefore R\,m_{m,j} = 0,4082 \text{ pav./dia}$$

Atribuindo uma despesa total $DespTotal_{m,j}$ de 1000 unidades monetárias *UM* para a atividade *j* no mesmo problema, se obteria a seguinte despesa diária:

$$\text{DespDiária}_{m,j} = 1000 / 49 \therefore \text{DespDiária}_{m,j} = 20{,}408 \text{ UM}$$

4.4.10. Expressões Gráficas Possíveis da Programação pelo Método Proposto

Com base nestes resultados numéricos gerados pelo método proposto, é possível construir inúmeras expressões gráficas, necessárias à melhor compreensão da programação em uma construção real. A figura 4.11 mostra o esboço de um tipo possível de expressão gráfica e sua relação com o edifício, transformando as atividades não repetitivas em barras de um gráfico de Gantt, e as atividades repetitivas em linhas de balanço.

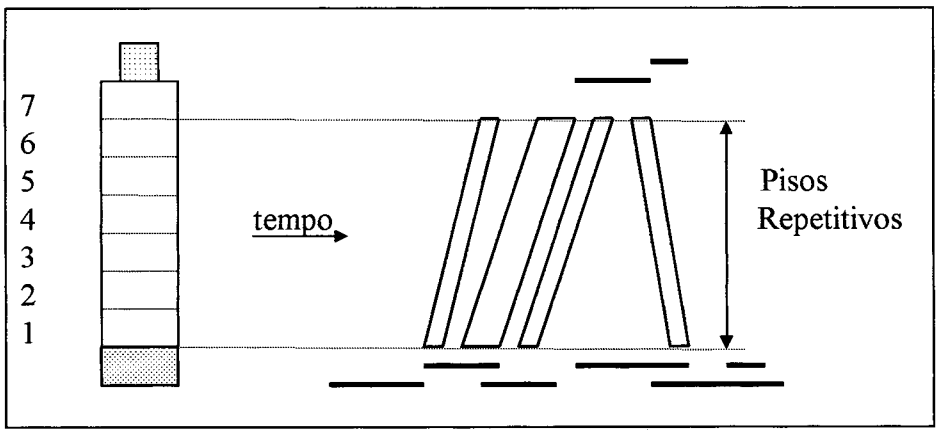


Fig. 4.11 - Esboço de uma possível expressão gráfica utilizando dados fornecidos pelo método proposto. Fonte: do autor.

4.5. A Estrutura Conceitual do Método Proposto

O método proposto pode ser dividido em dois módulos principais: (1) o Módulo de Busca (item 4.5.2), constituído pelo algoritmo genético, e (2) o Módulo de Programação (item 4.5.1), que incorpora o conhecimento específico do problema. O Módulo de Programação é composto por duas partes: o Esquema de Programação Serial e o Processador de Restrições. A estrutura conceitual deste sistema pode ser visualizada na figura 4.12 e cada uma das partes componentes está descrita nos próximos itens do capítulo.

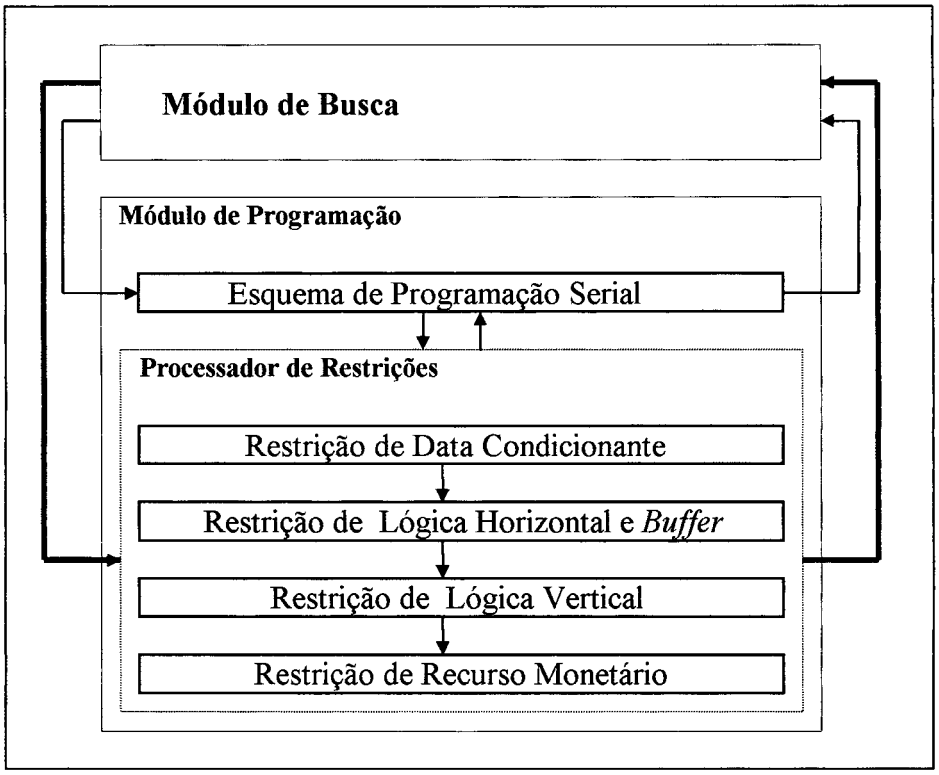


Fig. 4.12 - Componentes Gerais do Método Proposto: o Módulo de Busca e o Módulo de Programação. A seta fina indica o processo de geração da população inicial e a seta grossa indica o processo do cumprimento das restrições. Fonte: do autor.

4.5.1. O Módulo de Programação

Este módulo consiste no método de solução heurístico que incorpora os preceitos inerentes à classe dos Problemas de Programação de Recursos com Restrição de Recursos, bem como incorpora conhecimentos relativos à programação de projetos repetitivos de Construção Civil. Para realizar a alocação dos recursos, o Módulo de Programação utiliza o Esquema de Programação Serial (SSS - *Serial Schedule Scheme*) com um único passo, assim como adota geração randômica para definir a ordem de programação e o número de equipes para cada atividade.

O SSS visto no capítulo II, foi adaptado de forma a atender às peculiaridades do problema em evidência, bem como para atender ao sequenciamento aleatório do método proposto. O atendimento às restrições, consiste na chamada ao Processador de Restrições, referente à Restrição de Data Condicionante, Restrição de Lógica Horizontal e *Buffer*, Restrição de Mão-de-Obra e Restrição de Recurso Monetário. As datas de início e término da atividades, denominadas $Inicio_j$ e $Final_j$, vão sendo ajustadas progressivamente do primeiro ao último processador de restrição. A figura 4.13 mostra o procedimento principal.

O Módulo de Programação possui duas funções: a de formar a população inicial de programas viáveis, para ser manipulada pelo algoritmo genético do módulo de busca local; e a de realizar o cumprimento das restrições de todos os programas das

gerações posteriores. À partir da primeira iteração, o Módulo de Busca deixa de usar o Esquema de Programação Serial, mas continua a usar o Processador de Restrições para programar as recombinações decorrentes das aplicações dos operadores genéticos.

Esquema de Programação Serial

Inicialização
$S_1 := 1;$
$n := 2;$ $\{n = 1, \ldots, J$ *estágios*$\}$
$ModoProj_1 := 0;$
Enquanto $|S_n| < J$ ***faça*** *Estágio n*
Compute D_n
$J^* :=$ *random* $(j / j \in D_n);$
$NEq^* :=$ *random* $(NEq / NEq \in Neq_{m,j});$
Restrição de Data Condicionante;
Restrição de Lógica Horizontal e Buffer;
Restrição de Lógica Vertical;
Restrição de Recurso Monetário;
$\{Início_j, Final_j\}$
$S_n := S_n \cup \{j^*\};$
$n := n + 1;$
$PriorProj_n := j^*;$
$ModoProj_n := NEq^*;$
$PriorProj_J := J;$
$ModoProj_J := 0;$

Onde:

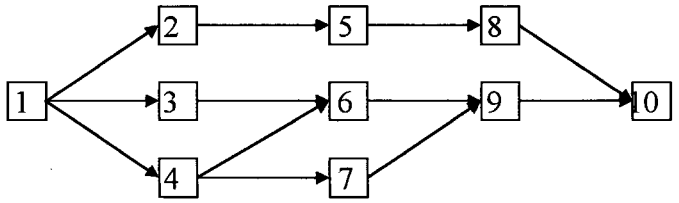
D_n : *Conjunto de Decisão;* $D_n := \{j / j \notin S_n, P_j \subseteq S_n\}$
$Início_j$: *Data programada para o início da atividade j;*
FinaC : Data programada para o término da atividade j;
$ModoProj_n$: *Lista de Modos das Atividades do Projeto;*
$NEq_{m,j}$: *Número de Equipes da atividade j no modo m;*
P_j:*Conjunto das Precedentes da Atividade j;*
$PriorProj_n$: *Lista de Priorização das Atividades do Projeto;*
S_n :*Conjunto das Atividades Programadas no estágio n;*

Fig. 4.13 - Esquema da Programação Serial (*SSS - Serial Schedule Scheme*) em Pseudocódigo. Adaptado de Kolisch (1996). Fonte: do autor.

4.5.1.1. A Priorização das Atividades

A forma tradicional do método serial utiliza uma regra de priorização para as atividades, que geralmente reside na atribuição de pesos. Fugindo a este desígnio,

adota-se neste trabalho um sistema de priorização randômica, respeitando-se as restrições de precedências estabelecidas no grafo. A idéia deste mecanismo pode ser visto na figura 4.14, onde mostra-se o grafo representativo de um projeto e duas priorizações viáveis *a* e *b*, as quais iniciam e finalizam sempre pelas atividades *dummies* (*1* e *10*). No início do processo da priorização *a*, poderiam ser escolhidas as atividades *2*, *3* ou *4* (a atividade *2* foi escolhida aleatoriamente); depois estava entre 3, 4 ou 5 (a atividade *5* foi escolhida aleatoriamente), e assim por diante. A figura mostra também uma priorização inviável, onde as atividade *6* e *8* rompem precedências.



Priorizações Viáveis: 1 - 2 - 5 - 8 - 4 - 3 - 7 - 6 - 9 - 10 (a)
1 - 3 - 4 - 6 - 2 - 7 - 5 - 8 - 9 - 10 (b)
Priorização inviável : 1 - 3 - **6** - 4 - 7 - 2 - **8** - 5 - 9 - 10 (c)

Fig 4.14 - Exemplos de Priorizações Viáveis e Inviáveis em um Esquema de Programação Serial. Em negrito, observam-se prioridades incorretas. Fonte: do autor.

4.5.1.2. Os Tipos de Recursos Diretamente Restringidos

O método proposto analisa a existência de soluções viáveis sujeitas à dois tipos de restrições diretas de recursos: (1) o recurso monetário previsto para cada período de tempo (geralmente o mês), e (2) o número máximo de equipes a ser alocado à cada atividade (restrição de espaço).

4.5.1.3. A Restrição de Espaço

A restrição de espaço diz respeito ao número máximo de equipes de trabalho, que o ambiente suporta para a realização de uma determinada atividade. Este dado é normalmente fruto da experiência do programador da obra, e entre outros fatores, depende da capacidade do posto de trabalho e do nível de interferência de uma equipe sobre outra. Em conseqüência, o número de equipes $Neq_{m,j}$ alocado para uma atividade deve ser menor ou igual ao número máximo de equipes $NEqMax_j$ passível de ser alocado àquele espaço:

$$NEq_{m,j} \leq NEqMax_j \qquad [9]$$

4.5.1.4. A Restrição de Lógica Horizontal

Para satisfazer às restrições horizontais, deve-se inicialmente considerar o tipo das atividades envolvidas. Se ambas não são repetitivas ou se uma delas é repetitiva, a situação enquadra-se em um dos três primeiros casos a seguir; mas, se ambas as atividades forem repetitivas, enquadra-se em um dos casos restantes. Neste processo são sempre comparadas duas atividades, a atividade atual *j* e uma atividade imediatamente predecessora *i*. Se houver mais de uma atividade predecessora, utiliza-se a menor data possível para o início de *j*, a qual satisfaz ao sincronismo final-início.

■ **Caso 1** - Figura 4.15

- A atividade *j* é não repetitiva;
- A atividade *i* é não repetitiva;

Este é o caso típico das redes PDM e CPM, assim como do Gráfico de Gantt. Ou seja, o início da atividade *j* é a primeira data após o término da atividade *i*. Na figura 4.15-a as atividades *i* e *j* encontram-se em regiões diferentes (abaixo da torre e acima da torre, respectivamente); na figura 4.15-b as atividades encontram-se na mesma região (acima da torre).

$$\text{Início}_j = \text{Final}_i + 1 \qquad [10]$$

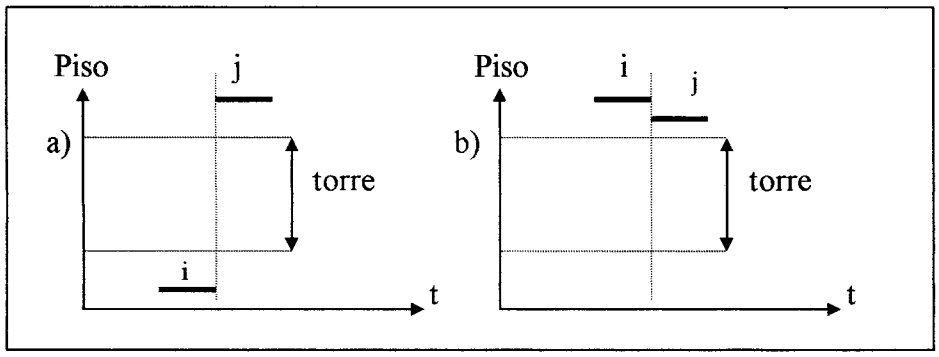


Fig 4.15 - Ligações entre Atividades Não Repetitivas: exemplificação do caso 1.
Fonte: do autor.

■ **Caso 2** - Figura 4.16

- A atividade *j* é não repetitiva;
- A atividade *i* é repetitiva;
- A atividade *i* possui sentido ascendente ou descendente;

Neste caso também existem duas possibilidades: (1) a atividade *i* é realizada em qualquer pavimento abaixo da torre; (2) a atividade *i* é realizada em

qualquer pavimento acima da torre. Em ambos os casos, a data de início de *j* é a primeira data após o término de *i*.

$$\text{Início}_j = \text{Final}_i + 1 \qquad [11]$$

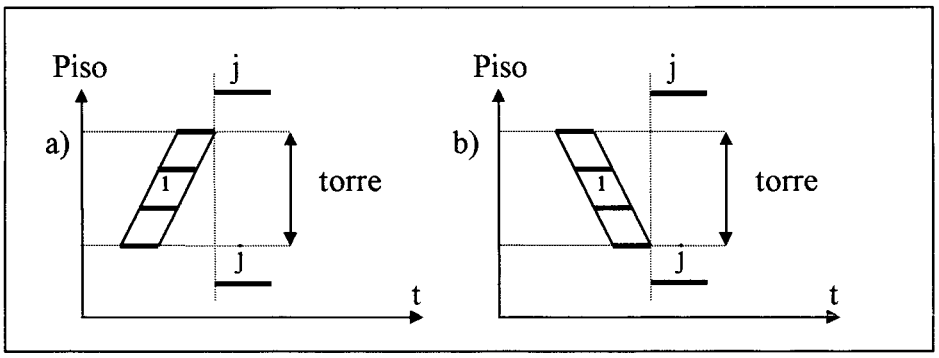


Fig 4.16 - Ligações entre Atividades de Naturezas Diferentes: exemplificação do caso 2. Fonte: do autor.

■ **Caso 3** - Figura 4.17

- A atividade *j* é repetitiva;
- A atividade *i* é não repetitiva;
- A atividade *j* possui sentido ascendente ou descendente;

Neste caso existem duas possibilidades: (1) a atividade *i* é realizada em qualquer pavimento abaixo da torre; (2) a atividade *i* é realizada em qualquer pavimento acima da torre. Em ambos os casos, a data de início de *j* é a primeira data após o término de *i*, independentemente do sentido de *j*.

$$\text{Início}_j = \text{Final}_i + 1 \qquad [12]$$

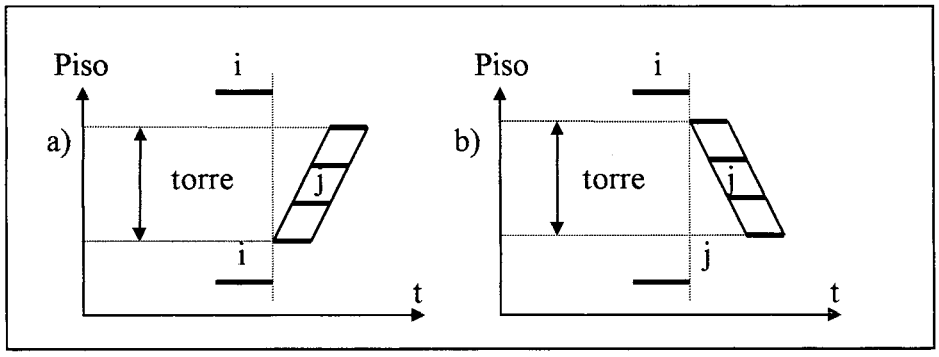


Fig 4.17 - Ligações entre Atividades de Naturezas Diferentes: exemplificação do caso 3. Fonte: do autor.

Se as atividades *i* e *j* forem repetitivas, existem 6 casos distintos que são mostrados a seguir. Para apresentá-los, torna-se necessário estabelecer o conceito de *piso crítico* [Thabet e Beliveau, 1994], o qual consiste no piso que precisa ser verificado para programar a atividade *j*; ao analisar o piso crítico, torna-se redundante verificar os demais. Em outras palavras, a data de início da atividade *j* é a primeira data que torna possível preservar o sincronismo final-início no piso crítico.

Nas atividades de sentido ascendente, o denominado pavimento repetitivo inicial PI_i corresponde ao pavimento em que a execução inicia. Para uniformizar esta convenção, nas atividades de sentido de execução descendente o PI_i corresponde ao último pavimento de execução desta atividade. Portanto, a referência consiste no pavimento e não na execução da atividade. A nomenclatura utilizada é a seguinte:

PI_i = Pavimento-tipo inicial da atividade *i*;
PF_i = Pavimento-tipo final da atividade *i*;
R_i = Ritmo da atividade *i*;

As figuras que mostram os casos *4* a *9* podem ser observadas a seguir, conjuntamente com as respectivas expressões.

■ **Caso 4** - Figura 4.18

- O ritmo da atividade *j* é maior ou igual ao da atividade *i*;
- As atividades *i* e *j* possuem o mesmo sentido *j*>0 e *i*>0;

Piso Crítico: o piso crítico é o último piso da atividade *i*.

$$Início_j = Final_i - (PF_j - PI_j) / Ritmo_j + 1 \qquad [13]$$

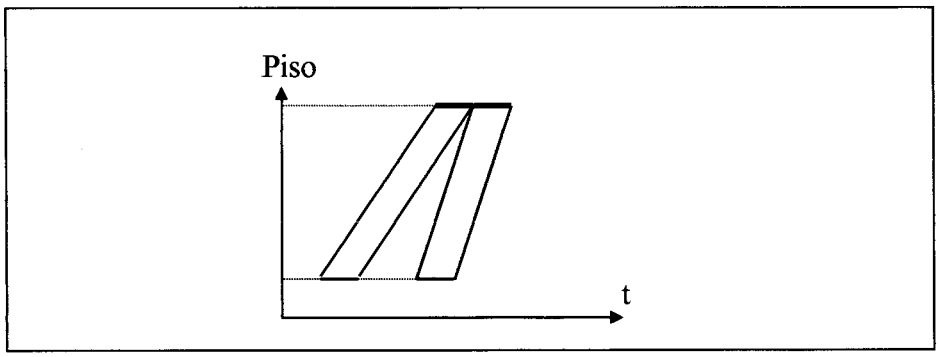


Fig 4.18 - Restrição Horizontal: exemplificação do caso 4. Fonte: do autor.

■ **Caso 5** - Figura 4.19

- O ritmo da atividade *j* é menor que o da atividade *i*;
- As atividades *i* e *j* possuem o mesmo sentido *j*>0 e *i*>0;

Piso Crítico: o piso crítico é o primeiro piso da atividade *i*.

$$\text{Início}_j = \text{Final}_i - (\text{PF}_i - \text{PI}_i) / \text{Ritmo}_i - (\text{PI}_i - \text{PI}_j) / \text{Ritmo}_j + 1 \quad [14]$$

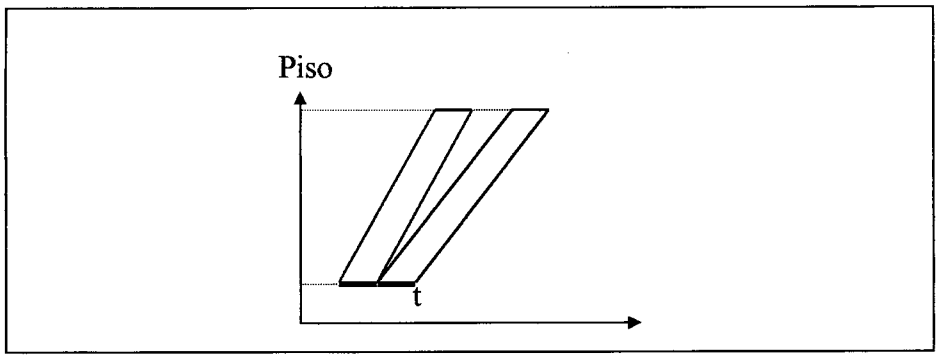


Fig 4.19 - Restrição Horizontal: exemplificação do caso 5. Fonte: do autor.

■ **Caso 6** - Figura 4.20

- As atividades *i* e *j* possuem os sentidos *j*>0 e *i*<0;

Piso Crítico: o piso crítico é o primeiro piso da atividade *j*.

$$\text{Início}_j = \text{Final}_i - (\text{PI}_j - \text{PI}_i) / \text{Ritmo}_i + 1 \quad [15]$$

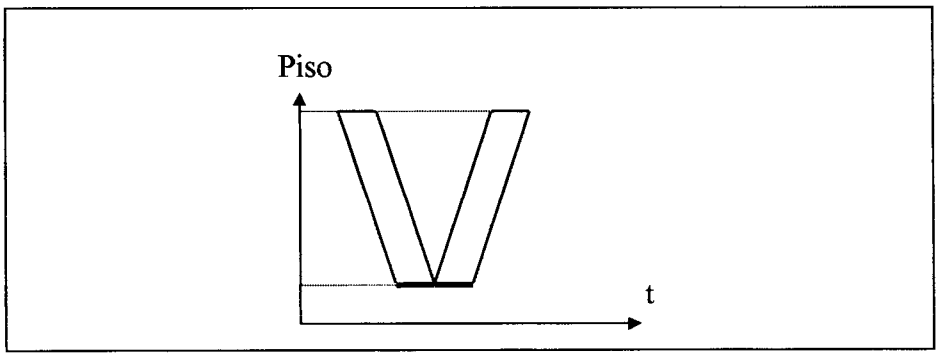


Fig 4.20 - Restrição Horizontal: exemplificação do caso 6. Fonte: do autor.

■ **Caso 7** - Figura 4.21

• As atividades *i* e *j* possuem os sentidos $j<0$ e $i>0$;

Piso Crítico: o piso crítico é o último piso da atividade *j*.

$$\text{Início}_j = \text{Final}_i - (\text{PF}_i - \text{PF}_j) / \text{Ritmo}_i + 1 \qquad [16]$$

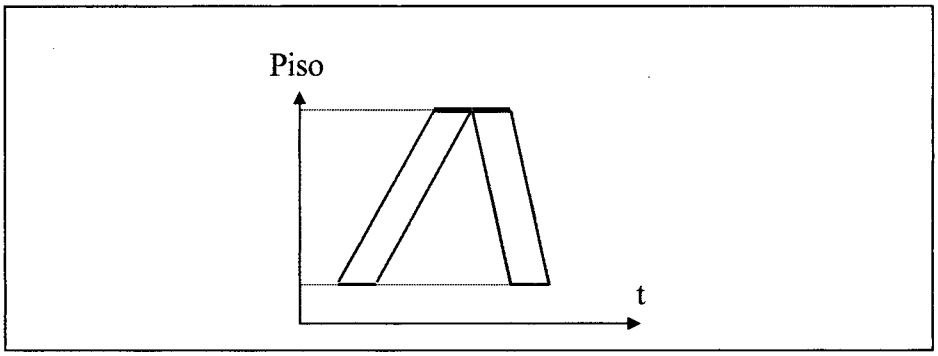


Fig 4.21 - Restrição Horizontal: exemplificação do caso 7. Fonte: do autor.

■ **Caso 8** - Figura 4.22

• O ritmo da atividade *j* é maior ou igual ao da atividade *i*;
• As atividades *i* e *j* possuem o mesmo sentido $j<0$ e $i<0$;

Piso Crítico: o piso crítico é o primeiro piso da atividade *i*.

$$\text{Início}_j = \text{Final}_i - (\text{PF}_j - \text{PI}_j) / \text{Ritmo}_j + 1 \qquad [17]$$

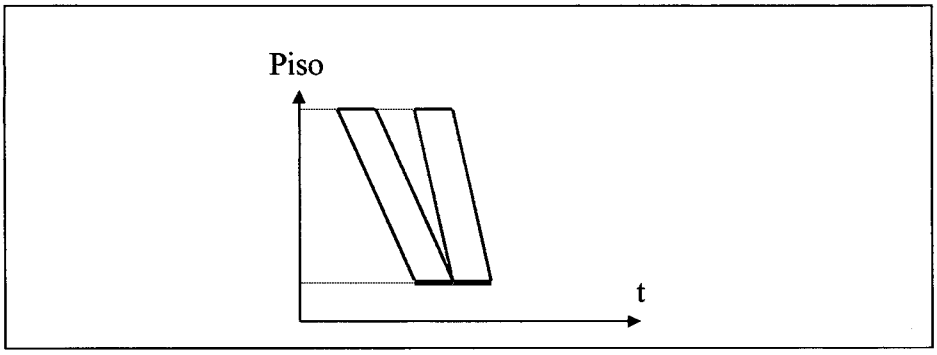


Fig 4.22 - Restrição Horizontal: exemplificação do caso 8. Fonte: do autor.

■ **Caso 9** - Figura 4.23

- O ritmo da atividade *j* é menor que o da atividade *i*;
- As atividades *i* e *j* possuem o mesmo sentido *j*<0 e *i*<0;

Piso Crítico: o piso crítico é o último piso da atividade *i*.

$$Início_j = Final_i - (PF_i - PI_i) / Ritmo_i - (PF_j - PF_i) / Ritmo_j + 1 \quad [18]$$

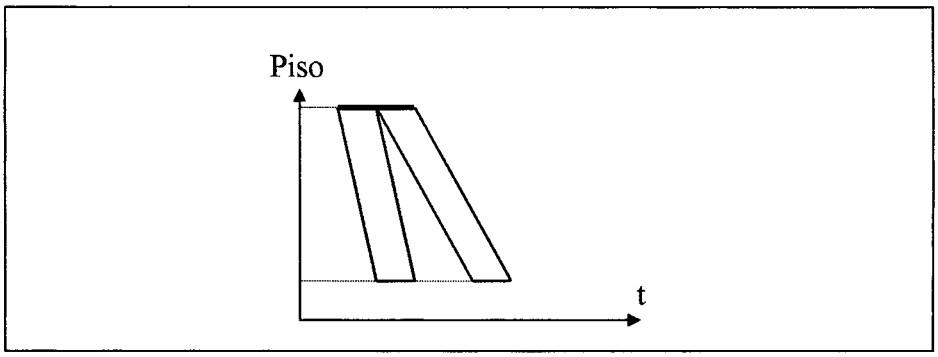


Fig 4.23 - Restrição Horizontal: exemplificação do caso 9. Fonte: do autor.

4.5.1.5. A Restrição de *Buffer*

A restrição de *buffer* também é um tipo de restrição na direção horizontal, e refere-se ao afastamento mínimo em unidades de tempo, entre o início da atividade a ser programada e o final da atividade antecessora, no pavimento crítico (figura 4.24). Os *buffers* existem para prevenir programas muito rígidos e com alta sensibilidade para atrasos em cada atividade, a fim de que um eventual atraso na atividade atual não implique no atraso das sucessoras [Selinger, 1980]. No modelo proposto, o *buffer* é incrementado como uma parcela na expressão matemática criada para a restrição de lógica horizontal.

$$Início_j = [Início\ por\ Restrição\ de\ Lógica\ Horizontal\ ] + buffer \quad [19]$$

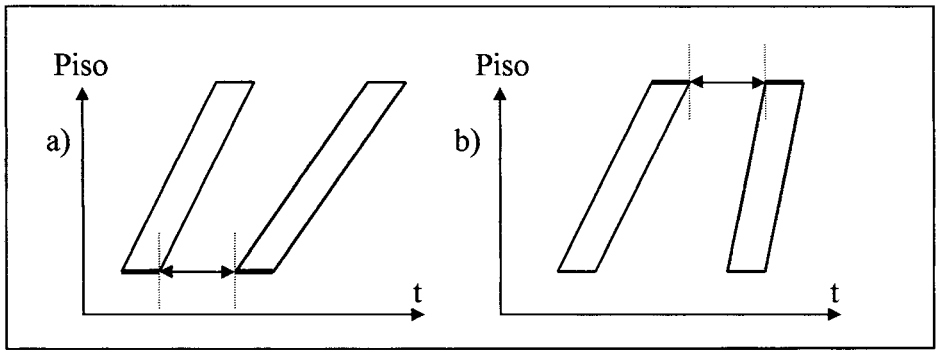


Fig. 4.24 - Restrição de *Buffer*: exemplificação sobre os casos 5 e 4 de lógica horizontal, respectivamente. Fonte: do autor.

4.5.1.6. A Restrição de Data Condicionante

Em muitas situações relacionadas à causas internas ou externas ao projeto, torna-se necessário impor uma data mínima para o início de uma atividade. A data da chegada de um recurso ou uma data politicamente ou estrategicamente interessante para a empresa, são exemplos práticos deste tipo de restrição $DataCond_j$.

$$\text{Início}_j \geq \text{DataCond}_j \qquad [20]$$

4.5.1.7. A Restrição de Lógica Vertical

Em alguma circunstância, pode ser necessário começar uma atividade repetitiva somente após o término da execução de uma atividade predecessora em um outro pavimento. Esta idéia compõe a restrição de lógica vertical (RV), a qual está relacionada a vários fatores, como por exemplo à dependência tecnológica entre as atividades, datas impostas ou restrições de segurança. Sob esta consideração, a primeira data possível para o início de uma atividade *j*, depois de concluídos RV_{ij} pavimentos da atividade *i*, pode ser calculada pela expressão a seguir:

$$\text{Início}_j = \text{Início}_i + (\text{RV}_{ij} * \text{dTipo}_{m,i}) \qquad [21]$$

$$\text{Com} \quad 1 \leq \text{RV}_{ij} \leq \text{Ntipo} \qquad [22]$$

Usando como exemplo uma atividade *j*, que deve iniciar somente quando o quarto pavimento de sua predecessora *i* estiver concluído. Considerando que a atividade *i* tenha início no dia *8* e sua duração $dTipo_{m,i}$ seja de *3* dias, o início de *j* é calculado da seguinte forma:

$$\text{Início}_j = 8 + (4 * 3) \quad \therefore \text{Início}_j = 20 \text{ dias}$$

A visualização gráfica da restrição de lógica vertical encontra-se na figura 4.25, na qual a atividade *j* só pode iniciar após *y* pavimentos da atividade *i* terem sido concluídos. Um ponto passível de dúvida é a semelhança entre a lógica vertical e o *buffer*; a diferença é que a primeira restrição refere-se ao afastamento que uma atividade precedente impõe sobre uma sucessora, em um mesmo piso, e a segunda refere-se ao afastamento que uma atividade precedente de um determinado piso impõe sobre o início da sucessora.

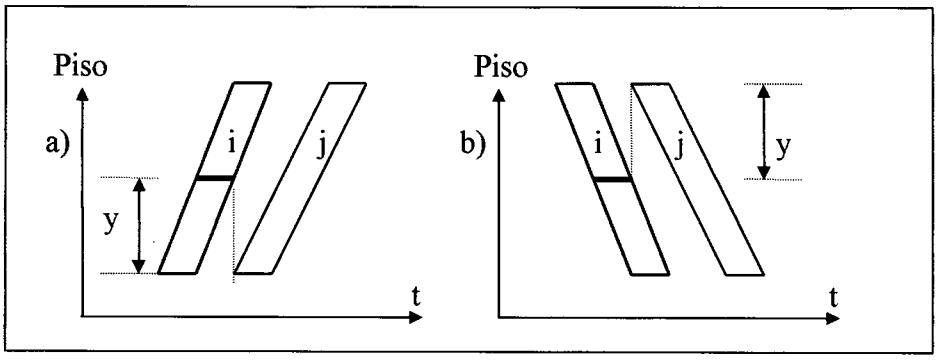


Fig. 4.25 - Exemplificação de Restrição de Lógica Vertical: a atividade *j* só pode iniciar após o pavimento *y* da atividade *i* ter sido concluído. Fonte: do autor.

A restrição de lógica vertical deve ser implementada após o processamento da restrição de data condicionante, do qual recebe valores iniciais para as variáveis $Início_j$ e $Final_j$, e pode ser codificada à partir do algoritmo mostrado na figura 4.26:

Lógica de Restrição Vertical

Inicialização
Início $_j$ = Início de j que satisfaz às restrições anteriores;
Final $_j$ = Final de j que satisfaz às restrições anteriores;
n := 1; {n = 1, ... , NPreced $_j$}
I := Início $_j$;
***Enquanto** / n ≤ NPreced $_j$ / **faça** Estágio n*
Se** RV $_{ij}$ <>0 **Então
*I = Início $_i$ - 1 + (RV $_{ij}$ * dTipo $_{m,i}$);*
Se** I > Início $_j$ **Então
Início $_j$ = I;
n := n + 1;

Onde:

RV$_{ij}$: Restrição vertical da atividade j devido à atividade i;
Dtipo $_{m,i}$: Duração de i com número de equipes NEq;
Npreced $_j$:Número de Precedentes da Atividade j;
Ntipo :Número de Pavimentos-tipo

Fig. 4.26 - Lógica de Restrição Vertical. Fonte: do autor.

4.5.1.8. A Restrição de Recurso Monetário

De posse da Curva de Agregação de Recurso Monetário Acumulado ou Curva "S" estabelecida para o projeto, esta restrição verifica quantas unidades de tempo

(dias) da atividade atual *j* podem ser utilizadas no mês *p* avaliado, de maneira que a despesa das atividades programadas apresente os menores desvios possíveis da curva de agregação. Se o número de dias for igual a zero, analisa-se o próximo mês, e este procedimento prossegue até que haja recurso disponível. Caso parte da atividade *j* esteja presente em outros meses, cada um deles também é verificado em relação ao desvio. Na figura 4.27 pode-se observar o algoritmo da restrição de recurso monetário, a qual necessita dos valores das variáveis $Início_j$ e $Final_j$ fornecidos pela restrição de lógica vertical. A despesa diária $DespDiária_{m,j}$ da atividade *j* sob um número de equipes $NEq_{m,j}$, é calculada pela expressão [7] mostrada anteriormente.

Lógica de Restrição do Recurso Monetário

Inicialização

$Início_j$ = *Início de j que satisfaz às restrições anteriores;*
$Final_j$ = *Final de j que satisfaz às restrições anteriores;*
p = *Mês em que ocorre a data* $Início_j$; $p = \{1,...,P\}$
$D:=0$ *{Despesa mensal relativa à atividade atual};*

Enquanto $(Final_j \geq InícioDoMês_p)$ ***ou*** $(p < P)$ ***faça***

$NDiasDisp=(CurvaSProj_p - DespAcumProj_p - D)/DespDiária_{m,j}$
$I :=$ *máx* $(Início_j, InícioDoMês_p)$;
$F :=$ *mín* $(Final_j, FinalDoMês_p)$;
Se $NDiasDisp_n < (F - I + 1)$ *e* $(NDiasDisp \geq 1)$ ***Então***
$Início_j := FinalDoMês_p - NDiasDisp + 1$;
$Final_j := Início_j + d_{m,j} - 1$;
$D := (FinalDoMês_p - Início_j + 1) * DespDiária_{m,j}$;
$p := p + 1$;
Senão Se $NDiasDisp < (F - I + 1)$ ***and*** $(NDiasDisp < 1)$ ***Então***
$p := p + 1$;
$Início_j := InícioDoMês_p$;
$Final_j := Início_j + d_{m,j} - 1$;
$D:=0$;
Senão Se $NDiasDisp \geq (F - I + 1)$ ***Então***
$D := D + (F - I + 1) * DespDiária_{m,j}$;
$p:=p+1$;

Onde:

$InícioDoMês_p$: *lista dos dias de início de cada mês;*
$FinalDoMês_p$: *lista dos dias de término de cada mês;*
NDiasDisp: dias disponíveis para a atividade em dado mês;
$CurvaSProj_p$: *recurso monetário acumulado previsto até o mês p;*
$DespAcumProj_p$: *recurso monetário já utilizado até o mês p;*
$DespDiária_{m,j}$: *despesa diária da atividade atual j no modo com modo m;*
$d_{m,j}$:*duração da atividade j com modo m;*

Fig. 4.27 - Restrição de Recurso Monetário em Pseudo-Código. Fonte: do autor.

A soma das despesas diretas relativas à programação das atividades, mês a mês, são armazenadas na lista $DespProj_p$, a qual consiste na curva de agregação real do programa. O algoritmo da figura 4.28 mostra o procedimento adotado.

Incremento das Despesas de Projeto

Inicialização
$Início_j$ = *Início da atividade;*
$Final_j$ = *Final da atividade;*
m_1 = *Mês onde a atividade j inicia;*
m_2 = *Mês onde a atividade j finaliza;*
Enquanto $(m_1 \leq InícioDoMês_p)$ ***faça***
$I := máx\ (Início_j, InícioDoMês_p)$;
$F := mín\ (Final_j, FinalDoMês_p)$;
$DespProj_p := DespProj_p + (F\text{-}I+1) * DespDiária_{m,j}$;
$m_1 := m_1 + 1$;

Onde:

$InícioDoMês_p$: *lista dos dias de início de cada mês;*
$FinalDoMês_p$: *lista dos dias de término de cada mês;*
$DespProj_p$: *recurso monetário já utilizado no mês p;*
$DespDiária m_j$: *despesa diária da atividade atual j com modo m;*

Fig. 4.28 - Incremento das Despesa da Atividade Atual no Vetor de Despesas Mensais do Projeto. Fonte: do autor.

4.5.2. O Módulo de Busca

O Módulo de Busca utiliza um Algoritmo Genético como método de busca das soluções. O modelo proposto reside na hibridização do algoritmo, a fim de obter maior eficiência e efetividade, mesmo que para isto haja sacrifício de robustez, relativamente à um AG puro.

4.5.2.1. Hibridização do Algoritmo Genético

Uma das principais vantagens dos Algoritmos Genéticos é a sua independência de domínio, visto que originalmente operam na codificação do problema e dispensam outros conhecimentos específicos do contexto do problema. Ao criar um algoritmo híbrido através da incorporação no modelo básico, de alguma heurística de domínio específico ou não, geralmente perde-se parte desta vantagem, mas é um procedimento válido quando o interesse é encontrar uma solução efetiva e eficiente ao

invés de estudar a performance do AG em si mesmo. De acordo com Davis (1991), existem três princípios para a hibridização:

■ Utilizar uma técnica de codificação que já seja conhecida e usada nos métodos usuais de solução do problema. Agindo deste modo, garantir-se-á duas vantagens: (1) a preservação dos conhecimentos do domínio do problema incorporados na codificação e (2) a ausência de impactos negativos resultantes de uma codificação nova ou desconhecida;

■ Hibridizar onde for possível, através da incorporação de elementos interessantes de outros algoritmos já experimentados na solução do problema. Como exemplo, se já existe um algoritmo rápido para o problema, ele pode funcionar como um gerador de soluções iniciais, as quais serão melhoradas com a aplicação dos Algoritmos Genéticos;

■ Adaptar os operadores genéticos - deve-se criar operadores genéticos compatíveis com o entendimento do problema, com a função dos operadores e com o tipo de codificação a ser utilizada no algoritmo híbrido.

4.5.2.2. Classificação das Restrições do Problema

O modelo proposto consta da programação de atividades interrelacionadas que compõem o projeto. Destaca-se inicialmente neste cenário a classificação e o conseqüente tratamento diferenciado dos dois tipos principais de restrições:

■ **Restrições Duras (*Hard Constraints*)** - Este tipo de restrição nunca pode ser violada ou relaxada, a qualquer preço. Por razões de eficiência, é vantajoso codificar a lógica para manter as restrições fortes, diretamente no mecanismo de alocação de recursos [Chan *et al.*, 1996], conforme foi visto no Módulo de Programação. Neste problema em evidência, os seis tipos de restrição abordados no item 4.5.1, constituem o tipo de restrição forte: a restrição de data condicionante; a restrição de espaço, a restrição de lógica horizontal, a restrição de lógica vertical, a restrição de *buffer* e a restrição de recurso monetário.

■ **Restrições Suaves (*Soft Constraints*)** - Este tipo de restrição pode sofrer relaxação, mas com penalidade de desempenho. Estas restrições estão incorporadas na função de avaliação do algoritmo. No problema em questão, os desvios entre a curva de agregação do recurso monetário disponível (não acumulado) e a curva de despesas do projeto constitui o tipo de restrição suave.

4.5.2.3. A Utilização dos Algoritmos Genéticos em Problemas com Restrições

Para utilizar Algoritmos Genéticos em um Problemas de Restrição (*CP - Constrained Problem*), deve-se transformar o *CP* em um Problema de Otimização com

Restrições (*COP - Constrained Optimization Problem*). Para realizar tal intento, deve-se definir um Algoritmo Genético que seja compatível com este tipo de otimização. Em conseqüência, surgem dois problemas fundamentais para serem resolvidos: definir um *COP* equivalente e definir um AG que seja aplicável, eficiente e efetivo para este COP.

Para criar um algoritmo que seja eficiente e efetivo, os Algoritmos Genéticos puros podem ser modificados e dois procedimentos sobressaem na literatura: (1) substituir a busca aleatória por um mecanismo que leve em conta as restrições, e (2) utilizar outras heurísticas para acelerar o processo de busca, a partir da geração de soluções iniciais. No presente trabalho utiliza-se ambos os procedimentos, através do Módulo de Programação, que fornece a população inicial e orienta o processo de busca segundo as restrições, evitando a busca *cega*.

4.5.2.4. A Função Objetivo

A transformação de um problema de restrições em um COP, pode ser feita através da utilização da Função *Penalty*. Procedendo assim, as restrições suaves são colocadas diretamente na função e cada violação de uma restrição. As diferenças ou desvios $(CurvaAgreg_p - DespProj_p)$ entre o recurso monetário previsto e o recurso monetário programado à cada mês, constituem as violações das restrições. As parcelas são elevadas ao quadrado para evitar que desvios de sinais contrários se anulem no somatório da função.

$$f = \sum_{p=1}^{P} (\text{CurvaAgreg}_p - \text{DespProj}_p)^2 \qquad [23]$$

Onde CurvaAgreg_p = Curva de Agregação do Recurso Monetário;
DespProj_p = Curva de Despesas do Programa;

Visando expressar a função objetivo f como um percentual do recurso monetário disponível total para o projeto, utiliza-se a forma:

$$f = (1/\sum_{p=1}^{P} \text{CurvaAgreg}_p) * \sum_{p=1}^{P} Abs\,(\text{CurvaAgreg}_p - \text{DespProj}_p) \qquad [24]$$

Onde o operador *Abs(x)* retorna o módulo ou valor absoluto da expressão matemática x.

4.5.2.5. A Estruturação do Cromossoma

A estruturação do Cromossoma é um dos passos mais importantes da elaboração de um algoritmo genético, pois dela depende a lógica de funcionamento e a definição dos operadores genéticos. Cada solução individual é representada por um cromossoma, o qual consiste basicamente de genes dispostos seqüencialmente. Dois atributos são associados à cada gene: sua posição e o conteúdo que codificam (vide figura 4.28).

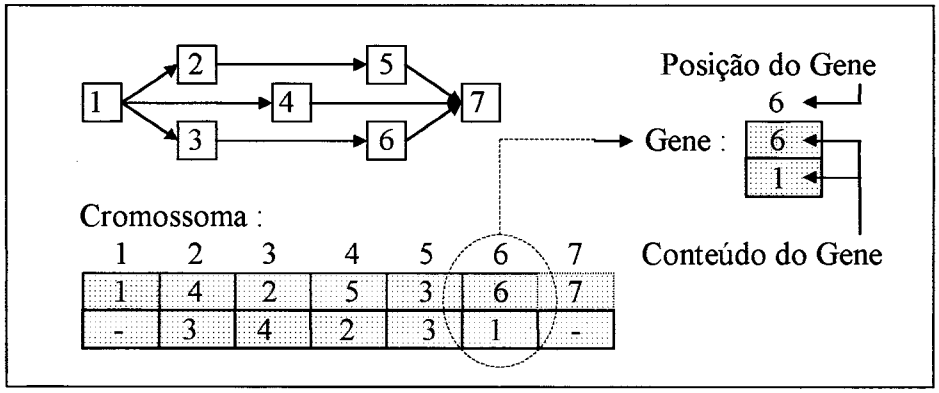


Fig. 4.29 - Representação do Cromossoma, exemplificado à partir de um grafo. Fonte: do autor.

Pode-se dividir esta estruturação em três partes distintas: (1) a escolha da variável de decisão que será codificada, (2) a definição do relacionamento entre a atividade e a posição do gene, (3) a forma de codificação para ser utilizada pelos valores dos genes.

A escolha imprópria da variável de decisão causa muitos retrabalhos e debilita o vínculo entre os esquemas bons e as realimentações positivas. Segundo Chan *et al.* (1996), na programação da construção civil duas alternativas são sugeridas: os genes representam as datas de início das atividades ou representam a prioridade com que se programa as atividades. A segunda opção é a mais recomendável, visto que na escolha da primeira, as relações de precedência são freqüentemente violadas, exigindo muitos consertos. O motivo é que as relações de precedência entre as atividades não são codificadas no cromossoma, fazendo com que o algoritmo genético tenha que *aprender* por exploração através das datas de início das atividades.

Adotando o procedimento de o conteúdo dos genes representar as prioridades das atividades, a posição relativa do gene no cromossoma não corresponde ao código numérico da atividade. E considerando-se que o problema abordado é de múltiplos modos, todo gene contém outra informação adicional associado à atividade que ele contém: o modo de execução programado. Uma importante vantagem desta representação, é que ela é a mesma utilizada pelo esquema de programação serial adotada no Módulo de Programação Serial. Assim, o Cromossoma na sua forma mais geral, corresponde à associação das listas $PriorProj_n$ e $ModoProj_n$ do item 4.5.1, caso

em que existem trechos do projeto modelados por atividades em série, bem como existem trechos constituídos por atividades em paralelo. Se a rede for inteiramente seriada, a primeira lista é redundante.

A codificação adotada é a de números inteiros representando cada atividade, de *1* a *N*, devido ser mais natural para o tipo de problema em evidência, que a tradicional codificação binária.

4.5.2.6. Níveis de Especialização do Cromossoma

No método proposto, o cromossoma está embutido em uma estrutura mais genérica, na qual existem níveis de especialização à partir de uma classe mais geral. O nível denominado *Gerações*, refere-se às iterações *g* do algoritmo, com $1 \leq g \leq NGer$, onde *NGer* é o número de gerações; o nível *Populações* refere-se aos indivíduos (cromossomas) *p* de cada Geração, com $1 \leq p \leq NIndPop$, onde *NIndPop* é o número de indivíduos na população; o nível *Indivíduos* diz respeito à cada um dos cromossomas de cada população, com $1 \leq i \leq N$, onde *N* é o número de atividades de cada indivíduo ou o número de atividades do projeto. Finalmente, o nível *Atividade* é o mais específico, e contém o código numérico da atividade, associado a um modo de produção. A figura 4.29 contém a esquematização desta estrutura.

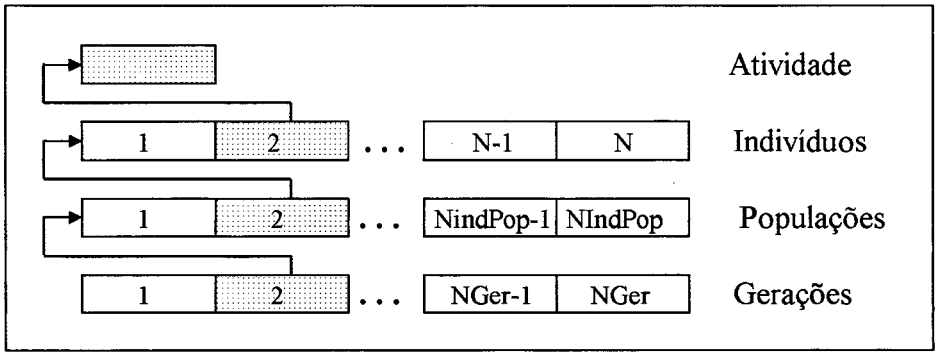


Fig. 4.30 - Níveis de Especialização do Cromossoma no Método Proposto. Fonte: do autor.

4.5.2.7. Geração da População Inicial

A geração da população inicial (fig. 4.30) é realizada pelo Módulo de Programação, com uma quantidade de indivíduos determinada por *NIndPop*. Após este passo, cada indivíduo da população é avaliada pela função *Penalty* do algoritmo genético, apresentada no item 4.5.2.4, e pelas expressões do item 4.5.2.8, a fim de que sejam escolhidos os pais que irão gerar descendentes para compor a nova população (1ª geração do algoritmo).

Geração da População Inicial

Inicialização
p = 0 ; p = {1,..., NindPop}
Enquanto** (p ≤ NIndPop) **faça
p :=p + 1;
Implemente o Módulo de Programação;

Onde:
NIndPop : Número de Indivíduos na População;

Fig. 4.31 - Geração da População Inicial. Fonte: do autor.

4.5.2.8. Métodos de Seleção Adotados

O método de seleção implica na escolha dos indivíduos que gerarão descendentes para a população, bem como na definição de quantos descendentes cada um destes indivíduos deverá gerar. Nos testes iniciais realizados com o protótipo computacional do modelo proposto, freqüentemente o algoritmo convergia prematuramente, devido às consideráveis diferenças de qualidade dos cromossomas (fato decorrente das restrições e dos múltiplos modos de produção envolvidos); por este motivo, optou-se por uma seleção que resulta em evolução mais lenta, mas que proporciona alta diversidade nas gerações iniciais: a Seleção Boltzmann. Para compreender o funcionamento deste método, é necessário antes abordar a teoria do *Simulated Annealing*.

- ***Simulated Annealing***

O *Simulated Annealing* (Temperamento Simulado) é uma técnica de Monte-Carlo usada para encontrar soluções em problemas de otimização. A técnica simula o resfriamento de átomos quentes vibrando. Quando átomos estão em alta temperatura, ficam livres para moverem-se por toda parte possível e tendem a moverem-se com movimentos randômicos. Mas, quando uma massa esfria, as ligações entre as partículas forçam os átomos em conjunto. Quando a massa está fria, não é possível o movimento, e a configuração é congelada. Se a massa é resfriada rapidamente, então a chance de obter uma solução de baixo custo é menor do que quando ela é resfriada vagarosamente (ou temperada).

Para uma dada temperatura, uma nova configuração de átomos é aceita se a energia do sistema é diminuída. Mas, se a energia é mais elevada, então a configuração

somente é aceita se a probabilidade de tal aumento é menor que a esperada para uma dada temperatura. Esta probabilidade é dada pela seguinte expressão:

$$P(\Delta E) = e^{-\Delta/KT}, \text{ onde K = Constante de Boltzmann.} \quad [25]$$

Os critérios de aceitação do *Simulated Annealing* são baseados na Física do Temperamento (*Physics of Annealing*). Muitos problemas de otimização podem ser considerados como um número de *objetos* (coisas), os quais precisam ser programados tal que uma função objetivo seja minimizada. Os átomos em vibração são substituídos por esses objetos, e o valor da função objetivo é substituída pela energia do sistema.

Um programa inicial é criado pela programação randômica dos objetos, e um custo inicial (Co) e temperatura (To) são computados. Permutações subsequentes são criadas pela escolha randômica de um número de objetos, rearranjando-os, e computando uma variação (troca) no custo (ΔC). Se $\Delta C \leq 0$, então a troca (variação) é aceita. Mas, se $\Delta C > 0$ a probabilidade de troca é calculada pela expressão:

$$P(\Delta C) = e^{-\Delta C/T} \quad [26]$$

Se a probabilidade é maior que um valor selecionado randômicamente no intervalo (0,1), então a troca (variação) é aceita. A técnica mais comum para escolher um número randômico é usar um pseudo-randômico uniformemente distribuído, variável sobre o intervalo de unidade. Após um número de permutações sucessivas a temperatura é diminuída para uma taxa de resfriamento (*Cooling Rate*) R, tal que $T_n = T_{n-1} * R$, onde $0 \leq R < 1$, e T é um número real.

Uma das vantagens do *Simulated Annealing* sobre outros algoritmos que pesquisam uma melhor solução (Algoritmos que escalam colinas = *hill climbing algorithms*), é que ele é menos provável de ser capturado em um local mínimo, porque o custo pode tanto aumentar quanto diminuir [Abramson, 1991].

Desde que Kirkpatrick *et al.* (1983), e Cerny (1985) introduziram o *Simulated Annealing* como um eficiente algoritmo para resolver problemas combinatoriais, muitas aplicações para vários problemas tem sido apresentados, como podem ser vistos em Van Laarhoven e Aarts (1987) e Aarts e Korst (1989); Osman e Potts (1989), Ogbu e Smith (1990), e Ishibushi *et al.* (1995), propuseram diferentes SA's para o problema de sequenciamento *flow shop.*

O *Simulated Annealing* tem sido muito pesquisado nos últimos anos. Johnston *et al.* (1989) fazem uma revisão de SA, incluindo uma analogia com a física do crescimento de cristais. Aarts e Korst (1989) provêem uma história do aparecimento da idéia principal nos anos cinqüenta e suas primeiras aplicações em problemas de otimização. Estas mesmas informações podem ser encontradas em Lundy e Mees (1986).

Muitos pesquisadores tem usado o SA para problemas de programação (*Scheduling*); na área de programação de máquinas, Matsuo, Suh e Sullivan (1989)

desenvolvem um SA para o problema Job Shop Geral; Eglese e Rand (1987) e Wright (1988) sobre *timetabling*; Osman e Potts (1989) e Ogbu e Smith (1990) desenvolveram algoritmos para minimizar o *makespan* em *flow shops*; e Jeffcoat e Bulfin (1993) desenvolvem um SA para o *Resource-Constrained Scheduling* .

■ ***Seleção Boltzmann***

Uma típica implementação para a Seleção Boltzmann [Mitchell, 1996] é atribuir à cada indivíduo *n* o valor esperado *VE (n)*.

$$VE(n) = \frac{e^{f(n)/T}}{\langle e^{f(n)/T} \rangle} \qquad [27]$$

Onde VE (n) = Valor Esperado do indivíduo *n*;
T = Temperatura;
f (n) é o *fitness* de *n*;
$\langle e^{f(n)/T} \rangle$ = Média de $e^{f(n)/T}$ sobre a população.

■ **Seleção pelo Giro da Roleta**

A Seleção pelo Giro da Roleta é o mecanismo auxiliar que efetivamente escolhe os indivíduos na população: à cada indivíduo é atribuída uma fatia de uma roleta circular, sendo o tamanho da fatia proporcional ao valor esperado *VE(n)* do indivíduo *n*. A roleta é girada e o indivíduo sorteado será escolhido para gerar descendentes. A figura 4.31 mostra este funcionamento.

Seleção pelo Giro da Roleta

Inicialização
n = 0 ; n = {1,..., NindPop}
Soma := 0;
Enquanto *(n ≤ NindPop)* ***faça***
n := n + 1;
Soma := Soma + VE (n);
r := random (0, Soma);

$$n := \{n' / \sum_{n=1}^{n'} VE(n) \geq r\}$$

Onde:

VE (n) : Valor esperado do indivíduo n;

Fig. 4.32 - Seleção pelo Giro da Roleta. Fonte: Goldberg, 1989.

4.5.2.9. Elitismo

O Elitismo é uma adição aos métodos de seleção que força os AG's a reterem o mesmo número de melhores indivíduos à cada geração, a fim de que não sejam destruídos pelos operadores de Cruzamento e Mutação, ou perdidos se não forem selecionados para reproduzir. É uma versão artificial da seleção natural enunciada por Darwin, com o objetivo de preservar as melhores soluções.

Em tese, este artifício não seria necessário, já que o algoritmo é evolutivo; ou seja, uma boa solução perdida em uma geração, possui grandes probabilidades de reaparecer nas futuras. O problema é que sempre existe a possibilidade de ela não reaparecer no número estabelecido de iterações do algoritmo. A taxa de elitismo *TElit*, presente na entrada de dados do algoritmo genético proposto, estabelece quantos indivíduos devem sobreviver ou ser replicados de uma geração para outra.

A figura 4.32 contém o procedimento para a implementação do Elitimo. Para a sua efetivação, é necessária a aplicação prévia de um algoritmo de ordenação sobre os indivíduos, de acordo com o valor do *fitness*.

Algoritmo da Replicação com Elitismo

Inicialização
    TElit : = Taxa de Elitismo;
    *EL : = TElit * NIndPop;*
Ordene Crescentemente os Indivíduos da População Antiga (fitness);
Cont : = NIndPop; {Contador}
Enquanto *(Cont ≤ EL)* ***Faça***
    Cont : = Cont – 1;
    Copie o Indivíduo Cont para a Nova População;

Onde:

NIndPop = Numero de Indivíduos na População

Fig. 4.33: Pseudo-Algoritmo Utilizado para a Reprodução com Elitismo. Fonte: do autor.

4.5.2.10. Cruzamento e Inversão

Utiliza-se dois operadores de cruzamento: o Operador LOX, que concentra-se na priorização das atividades, e o Operador de Modos, introduzido neste trabalho, com a finalidade de concentrar-se nos modos de produção das atividades.

Devido ao freqüente rompimento das relações de precedência quando se utiliza o Operador LOX, torna-se necessário o uso do Operador de Inversão para restabelecer estas ligações, e tornar viável o cromossoma. Por sua vez, o Operador de modos torna-se importante à medida em que o projeto possui poucas atividades em paralelo; no caso extremo de as atividades estarem todas em série, o Operador LOX deixa de ser utilizado.

■ Operador LOX e o Operador de Inversão

O Operador LOX gera garantidamente cromossomas legais, no sentido de não apresentar repetição de valores. A transferência das atividades que encontra-se entre os pontos de cruzamento para outras posições do cromossoma, assegura que nenhuma atividade será duplicada ou ausente nos descendentes. A maneira com que esta transferência acontece, de forma seqüenciada e ordenada, preserva tanto quanto possível as seqüências presentes nos pais.

A fim de mostrar o trabalho conjunto do LOX com a Inversão, utiliza-se o grafo da figura 4.33, em que existem *10* atividades em uma rede mista. Com base nesta representação pode-se construir os Cromossomas *S1* e *S2*, através do Esquema de Programação Serial, apresentando dois atributos: o código numérico da atividade e o modo de execução de cada atividade.

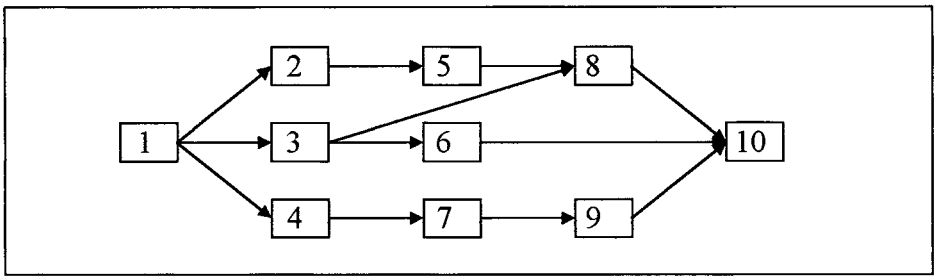


Fig. 4.34 - Grafo Atividade-no-nó para um Projeto de 8 Atividades (as atividades inicial e final são fantasmas). Fonte: do autor.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
S1:	1	3	2	4	5	8	7	6	9	10
modo:	-	2	1	5	1	2	2	1	3	-

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
S2:	1	4	7	9	3	6	2	5	8	10
modo:	-	1	3	2	1	2	4	3	1	-

Aplicando o Operador LOX nos cromossomas S1 e S2, sob os pontos de cruzamento *5* e *7*, se obtém o primeiro descendente através dos passos *a* e *b*:

S1 = 1 - * - * - 4 - [5 - 8 - 7] - * - 9 - 10 [a]

S1 = 1 - 4 - ***5*** - 8 - [3 - 6 - ***2***] - 7 - 9 - 10 [b]

O cromossoma resultante da aplicação do operador de cruzamento resultou em uma solução inviável, porque a atividade *5* situa-se antes de sua antecessora *2*. Assim, o passo *c* constitui a primeira inversão necessária:

S1 = 1 - 4 - 2 - ***8*** - 3 - 6 - ***5*** - 7 - 9 - 10 [c]

Continuando a verificação da esquerda para a direita, verifica-se que a atividade *8* também situa-se antes de sua antecessora *5*, situação que dá origem à uma nova inversão. Ao verificar os demais genes, não se encontra mais outra quebra de precedência. Portanto, o cromossoma viável é o seguinte:

filho1

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	4	2	5	3	6	8	7	9	10
-	5	4	1	1	2	2	2	3	-

O filho 2 é construído sob o mesmo procedimento, e pode ser visto a seguir. Finalmente, é importante observar que apesar das mudanças na priorização das atividades, os seus modos de execução não sofreram alterações.

filho2

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	4	7	3	2	5	9	6	8	10
-	1	2	1	4	1	2	2	2	-

■ Cruzamento de Modos

Este cruzamento origina-se do Cruzamento de Dois Pontos: o primeiro pai é replicado, dois pontos de cruzamento são escolhidos por sorteio e os modos das atividades entre pontos são substituídos pelos modos das mesmas atividades presentes no segundo pai. Neste operador, não é necessário recompor soluções, visto que a manipulação acontece somente sobre os modos das atividades. Como exemplo, usando os mesmos cromossomas *S1* e *S2* apresentados anteriormente e os mesmos pontos de cruzamento *6* e *9*, se obtém as seguintes soluções viáveis (o segundo filho é obtido analogamente ao primeiro):

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	3	2	4	5	8	7	6	9	10
-	2	1	5	1	1	3	2	2	-

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	4	7	9	3	6	2	5	8	10
-	1	3	2	1	1	1	1	2	-

4.5.2.11. Mutação

Os Operadores de Mutação usados neste trabalho são a Mutação *Scramble*, a Mutação de Posição, a Mutação de Ordem, e a Mutação de Modo. As aplicações incidem sobre o número da atividade e/ou sobre o modo de execução, conforme o caso. Os exemplos são fundamentados no grafo da figura 4.39, e aplicam-se sobre o seguinte Cromossoma S3:

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
S3:	1	3	4	2	5	7	6	8	9	10
Modos:	-	2	5	1	1	2	1	2	3	-

■ **Mutação *Scramble* (*Scramble Mutation*)**

A Mutação *Scramble* é aplicada somente sobre os modos de execução (a aplicação sobre a priorização das atividades, conduziria à quebras importantes de *schemata*). De acordo com seu mecanismo, escolhe-se aleatoriamente dois pontos {*5* e *7*}; considerando que cada atividade possui cinco modos de execução, pode-se sortear novos modos dentro destes limites {*2*, *4*, *1*}, que substituirão os modos atuais. O Cromossoma passa a ter a seguinte configuração:

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
S3:	1	3	4	2	5	7	6	8	9	10
Modos:	-	2	5	1	**2**	**4**	**1**	2	3	-

■ **Mutação Baseada na Posição (*Position-Based Mutation*)**

Este operador é usado sobre a priorização das atividades (Sua aplicação sobre os modos de execução conduz freqüentemente à situações impróprias, quando duas atividades possuem diferentes valores limites de modos. Este seria um problema contornável ao custo de um maior esforço computacional, mas é desnecessário resolvê-lo frente à existência dos outros operadores). Assim, ao sortear as posições *5* e *7* de *S3*, obtém-se o seguinte Cromossoma:

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	3	4	2	**6**	5	7	8	9	10
-	2	5	1	**1**	1	2	2	3	-

■ **Mutação Baseada na Ordem (*Order-Based Mutation*)**

Pelos mesmos motivos apresentados no operador de mutação anterior, a Mutação Baseada na Ordem da Mutação é apresentada somente sobre as priorizações das atividades. Usando as mesmas posições *5* e *7*, o cromossoma passa a ter a seguinte forma:

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	3	4	2	**5**	7	**6**	8	9	10
-	2	5	1	**1**	2	**1**	2	3	-

■ **Mutação de Modo**

A Mutação de Modo consiste na adaptação da Mutação Simples para o contexto da mudança do modo de execução de uma atividade, sem que haja alterações nas priorizações. O objetivo deste operador é criar diversidade através da menor perturbação possível no Cromossoma; e exatamente por este motivo, apenas uma atividade é envolvida. O procedimento é simples: escolhe-se uma posição randomicamente, e sorteia-se um novo modo possível para a atividade que ocupa àquela posição. Se a posição sorteada em *S3* fosse a *4*, um possível descendente poderia ser:

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	3	2	4	5	8	7	6	9	10
-	2	1	**2**	1	4	2	1	3	-

4.5.2.12. Tipos de Redes *Versus* Tipos de Operadores

No método proposto cada concepção de grafo para modelar um projeto, com atividades somente em série ou admitindo também atividades em paralelo, possui um conjunto peculiar mais adequado de operadores de cruzamento e de mutação.

As redes mistas, o caso mais comum, utilizam na operação de cruzamento os operadores LOX e Inversão, e na operação de mutação os operadores Mutação de Ordem, Mutação de Posição e Mutação de Modo. Com o objetivo de não degenerar os cromossomas, somente uma mutação é realizada em cada seleção: objetivando perturbar a priorização pode-se realizar mudanças na ordem ou na posição de um gene, ou objetivando perturbar o modo de produção, pode-se realizar mudanças localizadas no modo de uma atividade. A presença da Mutação de Modo acontece porque os outros operadores utilizados não realizam mudanças no modo de uma mesma atividade; por exemplo, quando são trocadas posições em uma mutação de posição, os modos de cada atividade continuam os mesmos. Esta foi uma necessidade percebida nas primeiras

experimentações do método proposto, frente à expressiva quantidade de modos que cada atividade potencialmente possui.

Considerando-se uma taxa de decisão *Tdecisao* para decidir se a mutação vai ser feita sobre a priorização ou sobre os modos das atividade, o esquema apresentado para implementar as operações genéticas pode ser visto na figura 4.34 A variável denominada *Campo* de um alelo, diz respeito ao segmento do cromossoma em que não existe nenhum precedente da atividade em evidência.

Esquema de Operação Genética para Redes Mistas

Se *r < TCruzLOX* ***Então***
    LOX;
Se *r > Tdecisao* ***Então***
    Se *r < TmutPriorizacao* ***Então***
        Se *o campo de P1 envolve P2* ***Então***
            Se *o campo de P2 envolve P1* ***Então***
                Mutação Ordem
            Senão *Mutação Posição*
Senão Se *r < TmutModo* ***Então***
    Mutação Modo;

Onde:

P1 e P2: Posições sorteadas para a mutação;
TCruzLOX: Taxa ou Probabilidade de Realização do LOX;
TmutPriorizacao: Taxa de Realização da Mutação de Posição
E Mutação de Ordem;
Tmut Modo: Taxa de Realização da Mutação de Modo;
r : número randômico;

Fig. 4.35 – Esquema de Operações Genéticas para Redes Mistas. Fonte: do autor.

Quando o grafo possui somente atividades em série, não é possível realizar trocas de prioridades entre as atividades do cromossoma. Portanto, a utilização do LOX e da Inversão resultam em esforço computacional desnecessário (embora não gerem cromossomas inviáveis); neste caso, a utilização do Cruzamento de Modo é mais adequada. Pelo mesmo motivo, não é possível a utilização dos operadores Mutação de Ordem e Mutação de Posição, e valem somente a Mutação de Modo (a qual causa mudanças mínimas no cromossoma) e a Mutação *Scramble* (a qual causa mudanças maiores no cromossoma). O esquema referente à este contexto pode ser encontrado na figura 4.35.

Esquema de Operação Genética para Redes Seriadas

Se *r < TcruzModo* ***Então***
Cruzamento de Modo;
Se *r > Tdecisao* ***Então***
Se *r < TmutScr* ***Então***
Mutação Scramble
Senão Se *r < TmutModo* ***Então***
Mutação Modo;

Onde:

TCruzModo: Taxa ou Probabilidade de Realização do LOX;
TmutScr: Taxa de Realização da Mutação Scramble;
Tmut Modo: Taxa de Realização da Mutação de Modo;
r : número randômico;

Fig. 4.36 – Esquema de Operações Genéticas para Redes Seriadas. Fonte: do autor.

Em resumo, os tipos de operadores genéticos utilizados em cada tipo de rede, podem ser vistos na figura 4.36.

Tipo de Rede:	Operadores Usados:	Probabilidade:
Linear	Cruzamento de Modo	TCruzModo
	Mutação *Scramble*	TMutScr
	Mutação de Modo	TMutModo
Mista	LOX e Inversão	TCruzLOX
	Mutação de Ordem	TmutPriorizacao
	Mutação de Posição	TMutPriorizacao
	Mutação de Modo	TMutModo

Tab. 4.1 - Tabela dos Tipos de Rede *Versus* Operadores do Modelo Proposto. Fonte: do autor.

4.5.2.13. A Verificação do Tipo de Rede

A verificação algorítmica do tipo de rede pode ser feita através de um algoritmo introduzido neste trabalho, cuja finalidade é listar as atividades paralelas do grafo que modela um projeto; se o grafo é linear, o tamanho da lista resultante denominada *AtividadesParalelas* é zero. Considerando que as relações de precedência não mudam durante o processamento da programação, esta lista é uma característica do grafo, e pode ser calculada na inicialização dos dados.

A geração de *AtividadesParalelas* baseia-se na teoria dos fluxos em redes (*Flows in Networks*) [Marshall, 1971; Furtado, 1973; Netto, 1979], da seguinte forma:

Seja o Grafo dirigido $G=(p,a)$ (vide Capítulo II) representativo do projeto, sem arcos redundantes por dois motivos: (1) eles atribuem a condição de paralelismo à atividades que na verdade são seriadas, conforme pode-se visualizar na figura 4.40, e (2) eles criam operações desnecessárias ao algoritmo Genético, tal como verificações desnecessárias de precedências.

O nó *1* é denominado *fonte* e o nó *N* é denominado *ralo*; cria-se um arco fictício que sai do nó *N* e liga-se ao nó *1*. Todo arco possui uma grandeza associada denominada capacidade $C_{ij}>0$, onde *i* é o nó inicial e *j* é o nó final; a capacidade de todos os arcos é considerada constante e igual à do arco fictício a_{1N}. Para todo arco $a_{i,j}$ do grafo, existe um número $0< x_{ij} \leq C_{ij}$, chamado fluxo. As expressões matemáticas podem ser observadas a seguir, e um exemplo pode ser encontrado na figura 4.37.

$$\sum_{k=1}^{ns_j} X_{jk} - \sum_{i=1}^{np_j} X_{ij} = 0, \text{ com } j \neq 1,N \text{ (Conservação dos Fluxos)} \quad [28]$$

$$f_j = \sum_{i=1}^{ns_j} X_{ij}, \text{ com } i, j \in N \quad [29]$$

$$X_{jk} = f_j / ns_j \quad [30]$$

Onde f_j = fluxo que chega ao nó *j*;
ns_j = Número de Arcos Sucessores de *j*;
np_j = Número de Arcos Precedentes de *j*;
X_{ij} = fluxo que percorre o Arco $a_{i,j}$;

Logo, um nó *p* é : { Paralelo se $f_j < f^*$; Seriado se $f_j = f^*$

Onde: f^* = Fluxo que Transita no Arco Fictício $A_{1,N}$

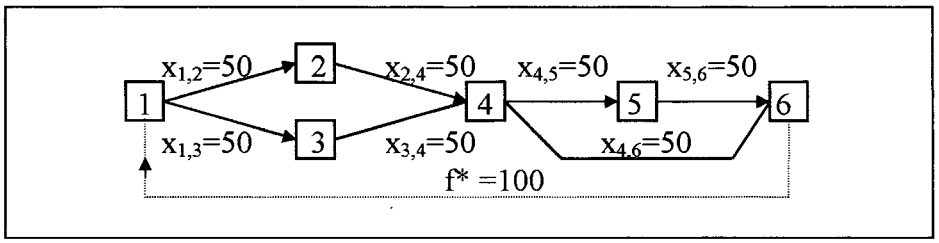


Fig. 4.37 - Exemplo do Conceito de Paralelismo entre os Nós de um Grafo: as atividades 2 e 3 são paralelas porque possuem o montante de fluxos $f_j=50<f^*=100$; a atividade 5 recebeu falsamente esta atribuição devido à existência do arco redundante $X_{4,6}$, que deve ser eliminado. Fonte: do autor.

A lista *AtividadesParalelas* possui outra finalidade igualmente importante para o método de solução apresentado: a de fornecer aos operadores Mutação de Ordem e Mutação de Posição, o conjunto das atividades passíveis de serem manipuladas (atividades paralelas). Mas, o fato de uma atividade ser um elemento da lista, não significa que ela seja paralela às demais, e sim que existe pelo menos uma outra que lhe seja paralela nesta lista. A necessidade de utilizar este artifício no método proposto, é devido à significância do número de atividades dispostas em série em projetos de construção civil.

4.5.2.14. Estrutura Básica do Algoritmo Genético

O algoritmo genético utilizado no método proposto pode ser descrito através dos passos genéricos apresentados na figura 4.38, fundamentados por Davis (1991):

Algoritmo Genético

Inicialização
Gere a População Inicial {de n Indivíduos};
Avalie o Fitness de cada Indivíduo;
***Enquanto** a Condição de Parada não for Verdadeira*
Aplique o Elitismo;
***Repita até** que a nova População esteja Completa*
Selecione dois Indivíduos da Geração;
Aplique Cruzamento para gerar dois Descendentes;
Aplique Inversão se Necessário;
Aplique Mutação nos Descendentes;
Aplique os Processadores de Restrições;
Avalie o Fitness dos Descendentes;
Coloque os Descendentes na nova População;
Geração Atual := Nova População;

Fig. 4.38 - Estrutura Básica do Algoritmo Genético, em Pseudo-Código. Fonte: Goldberg, 1989.

4.6. Conclusão

A programação de edifícios dotados de múltiplos pavimentos-tipo, visando minimizar as diferenças entre as curvas de agregação de recursos monetários prevista e programada, sujeita às restrições de data condicionante, lógica horizontal e *buffer*, restrições de lógica vertical, restrição de mão-de-obra e restrição de recurso monetário, constitui um problema de difícil resolução. A presença das atividades repetitivas, assim como a conseqüente necessidade de maximizar a continuidade das equipes ao longo dos pavimentos-tipo, motivaram a necessidade de integrar um modelo de grafo e um método de programação de projetos lineares, para representar o projeto.

Face ao grande número de atividades, variáveis e restrições, houve a necessidade de criar uma heurística de busca, que incorporasse o conhecimento relativo ao problema na geração da população inicial e no processo de alocação de recursos. Na elaboração do algoritmo, houve também a necessidade de inserir um mecanismo que impedisse a convergência prematura, e neste contexto, foi utilizada a Seleção de Boltzmann, baseada no *Simulated Annealing*. Por sua vez, o operador de cruzamento não poderia criar soluções inviáveis, tanto no sentido de repetir atividades, como no sentido de quebrar precedências; assim foi escolhido o *Linear Order Crossover* associado com o Operador de Inversão. Houve também a necessidade de adaptar operadores para manipular especificamente os modos de execução das atividades, daí surgirem o Cruzamento de Modo e a Mutação de Modo.

Em contrapartida às limitações impostas pela complexidade do problema, existem três artifícios para melhorar e/ou criar novos programas: (1) a maioria das atividades pode ser programada de vários modos de produção, onde cada modo significa o incremento de uma equipe de trabalho, respeitando-se as restrições de espaço; (2) a possibilidade de estabelecer várias seqüências de priorização à programação, dentre grupos de atividades paralelas; e (3) as atividades podem ter sua data de início deslocada para frente ou para trás na escala de tempo, desde que não implique na quebra de restrições.

CAPÍTULO V

EXPERIMENTOS E RESULTADOS

5.1. Introdução

Este capítulo apresenta as características principais dos problemas-teste utilizados, bem como apresenta os resultados dos experimentos realizados, os quais objetivam proporcionar indicativos sobre o desempenho do método proposto, frente ao problema de programação em evidência. Nesta análise, evidenciam-se as seguintes medidas: tamanho do problema, estrutura e complexidade, tempo de processamento, e resultado da função objetivo.

5.2. Os Atributos de um Problema de Programação de Projetos

Um problema de programação de projetos possui vários atributos, dentre os quais pode-se listar:

- Número de Atividades;
- Número de Arcos;
- Número de Modos por Atividade;
- Horizonte de tempo;
- Nível de disponibilidade de recursos;
- Perfis das Disponibilidades de Recursos;

Os valores numéricos de um determinado atributo interferem de maneira mais ou menos significante no desempenho do algoritmo de solução: o nível de dificuldade aumenta com o maior número de atividades, arcos, modos e horizonte de tempo, bem como aumenta quando a disponibilidade de recursos é restrita e seu perfil é irregular.

5.3. A Definição dos Experimentos

Em geral, o procedimento adotado para determinar o desempenho de um método de solução, consiste em testá-lo em problemas cujas redes de atividades possuem diversos tamanhos, estruturas e parâmetros [Demeulemeester *et al.*, 1993]. O tamanho de uma rede diz respeito ao número de arcos e de nós; uma estrutura em particular consiste em um contexto único destes arcos em relação aos nós, dentre o universo existente de possibilidades; e os parâmetros são grandezas que dependem da natureza do problema de decisão. Concluídos os experimentos, pode-se comparar os resultados

obtidos pelo método em evidência, com os resultados de outros métodos já reconhecidos pela literatura.

No presente trabalho de tese, o objetivo dos experimentos é proporcionar indicativos sobre o desempenho do método proposto. Para planejá-los, quatro fatores foram considerados fundamentais: (1) o domínio da aplicação é restrito à programação de projetos de construção de edifícios dotados de múltiplos pavimentos-tipo; (2) na literatura não existe outro método que seja destinado à otimizar este problema, tal como apresentado no capítulo anterior, o qual forneça resultados para realizar testes comparativos; (3) os grafos representativos deste tipo de projeto possuem um padrão estrutural significativo, já que o tipo e o interrelacionamento das atividades apresenta poucas variações; (4) as variações na concepção dos grafos do projetos são mais expressivas em variáveis como a quantidade de trabalho relativo à cada atividade, o número de pavimentos do edifício, o número máximo de equipes por atividade, e a disponibilidade de recursos. A análise destes fatores foi determinante para concluir que para atingir o objetivo almejado, os experimentos poderiam ser realizados com base em problemas reais, sem a necessidade de geração randômica.

Os problemas-teste foram extraídos de trabalhos acadêmicos do Programa de Pós-graduação em Engenharia de Produção e Sistemas da Universidade Federal de Santa Catarina, cujo conteúdo reside na programação de obras reais. Desta maneira, foram escolhidos quatro projetos, conservando os dados originais, e complementando-os quando necessário para a compatibilidade com os requerimentos do método proposto. Destes projetos originais, foram criados mais quatro problemas com objetivos definidos de experimentar a compressão do tempo (problema 5), experimentar projetos de tamanho pequeno com variação nas condições de contorno (Problemas 6 e 7), e experimentar um projeto com atividades seriadas (Problema 8).

A lógica dos experimentos consiste em assumir que os modos de produção e as durações adotadas nos trabalhos são as que melhor representam aos interesses das empresas construtoras, e que o perfil das despesas diretas das atividades programadas constituem a própria disponibilidade ou previsão de recurso monetário mensal. Sob este raciocínio, o valor ótimo da função objetivo é zero e acontece quando um programa gera uma curva de despesas igual à curva de disponibilidade de recursos monetários.

Os testes foram estabelecidos de modo a medir a eficiência (tempo de processamento computacional) e a efetividade, a qual denota a medida de qualidade dos resultados obtidos pelo método proposto. Considerando que o desvio mínimo entre as duas curvas de recursos monetários é zero, e que o desvio máximo corresponde à despesa direta total das atividades que compõem o grafo, a medida da efetividade de uma solução pode ser expressa através da relação entre o seu valor e a somatória das despesas diretas do projeto, em valores percentuais.

Devido à natureza probabilística dos Algoritmos Genéticos, para cada problema-teste foram realizados trinta experimentos através do protótipo computacional do método proposto. Desta maneira, obteve-se um total de 240 experimentos e o melhor

resultado de cada série foi utilizado para a medição do desempenho. Os dados pertencentes à cada problema podem ser encontrados no anexo deste trabalho.

5.4. Características Básicas dos Problemas-teste

As características consideradas básicas para caracterizar os problemas-teste são o número de atividades ou número de nós do grafo, o número de atividades repetitivas, e o número de interligações ou arcos. Todos estes dados podem ser vistos na tabela 5.1.

Número do Problema	Número de Atividades (*)	Número de Atividades Repetitivas	Número de Arcos
1	65	63	100
2	52	52	71
3	45	37	58
4	32	32	40
5	45	37	58
6	12	12	17
7	8	8	12
8	8	8	9

Tab. 5.1 - Características Básicas dos Problemas-Teste. (*) excluindo as fictícias.

5.5. As Medidas de Complexidade dos Problemas-Teste

Com o objetivo de relacionar a medida de efetividade do método proposto com a complexidade dos problemas-teste, evidenciou-se a necessidade de pesquisar índices que medem esta grandeza na literatura. As pesquisas de interesse estão ligadas à linha da programação de redes de atividades, ao problema do balanceamento de linhas de produção e ao problema da programação de máquinas.

Inúmeros trabalhos tem sido dedicados ao tema da complexidade no contexto das redes de atividades, dentre os quais destacam-se Pascoe (1966), Davis (1974, 1975), Kaimann (1974), Patterson (1976), Cooper (1976), Thesen (1977, *apud* Elmaghraby e Herroelen, 1980), e Elmaghraby e Herroelen (1980), cujas medidas apresentadas diferenciam-se essencialmente pela ênfase em uma determinada característica do problema geral. Segundo Elmaghraby e Herroelen (1980), uma medida de complexidade destina-se principalmente à: (1) prever o tempo de processamento requerido para a codificação computacional em um determinada configuração de *hardware*, e/ou (2) comparar dois ou mais algoritmos que possuam a mesma finalidade. No Presente trabalho, a medida de complexidade é utilizada explicar o maior/menor desempenho do algoritmo frente à cada problema-teste.

Devido ao fato de na literatura não existir uma medida inquestionável para medir a complexidade do RCPSP (*Resource Constrained Project Scheduling Problem*) sob múltiplos modos [Elmaghraby e Herroelen, 1980], optou-se pela utilização de dois índices: a densidade ou Coeficiente de Complexidade em Redes CNC (*Coefficient of Network Complexity*) [Pascoe, 1966], e a Densidade Média por Atividade ou AAD (*Average Activity Density*) [Patterson, 1976]. Estas medidas diferem-se das medidas adotadas na Teoria da Complexidade Combinatorial (*theory of combinatorial complexity*), devido ao fato de possuirem o objetivo mais específico de isolar os fatores que determinam o esforço computacional.

Sob o ponto de vista do CNC, a complexidade de uma rede é medida pela relação entre o número de arcos *A* e o número de nós *N*. Uma rede com *N* nós pode ter de [*N-1*] até [(*N-1*)**N/2*] arcos [Demeulemeester *et al.* ,1993]; portanto, a densidade varia de (*N-1*)/*N* até ((*N*-1)**N/2*))/*N*. Como exemplo, o problema *3* possui *34* nós (sendo dois fictícios) e *40* arcos, resultando em uma densidade de *1,176*, a qual situa-se em uma posição significativamente abaixo da média *8,74*, entre *33/34=0,97* e *33*17/34=16,5*. Em geral, os grafos representativos de projetos de Construção Civil possuem baixa densidade, visto que neles existem trechos com atividades em série e muitos dos arcos potenciais são redundantes (toda dependência indireta é redundante). Esta é uma característica que resulta na redução do esforço computacional requerido pelo processo de alocação de recursos, considerando que este esforço aumenta com o número de precedentes de cada atividade.

$$CNC = A / N \qquad [1]$$

Onde A = Número de Arcos;
N = Número de Nós;

O AAD considera que a complexidade pode ser obtida através do número de sucessores e de predecessores de cada atividade. Esta medida baseia-se na *T-density* proposta por Johnson [*apud* Elmaghraby e Herroelen, 1980], e conforme a expressão *3* aplicada no problema *3*, o valor do AAD é igual à (*2+1+1*)/*34=0,117*, correspondente às atividades *11*, *21* e *32*.

Seja o *Total Activity Density* ou *T-density*:

$$TAD = \sum_{N} \text{máx} \{ 0, NP - NS \} \qquad [2]$$

Onde NP = Número de Predecessores da Atividade;
NS = Número de Sucessores da Atividade;

A *Average Activity Density* consiste em:

AAD = *T-Density* / N [3]

Onde N = Número de Nós;

Na tabela 5.2 encontram-se os resultados das aplicações do CNC e do AAD para cada um dos problemas-teste.

Número do Problema	Densidade CNC	T-density Médio AAD
1	1,4925	0,4030
2	1,3148	0,2778
3 e 5	1,2340	0,1915
4	1,1765	0,1176
6	1,2143	0,2143
7	1,2000	0,3000
8	0,9000	0,0000

Tab. 5.2 - Medição da Complexidade dos Problemas-Teste.

5.6. Valores Limites das Características dos Problemas-Teste

Os valores máximo e mínimo das tabelas 5.1 e 5.2 encontram-se na tabela 5.3, com o objetivo de caracterizar o conjunto dos problemas-teste.

Características de Modelagem	Mínimo	Máximo
Número de Atividades	8	65
Número de Atividades Repetitivas	8	63
Número de Arcos	9	100
Densidade (CNC)	0,9000	1,4925
T-density (AAD)	0,0000	0,4030

Tab. 5.3 - Resumo das Características dos Problemas-Teste em seus Valores Limites.

5.7. Características da Implementação Computacional do Protótipo

O protótipo desenvolvido para testes foi implementado em um microcomputador Pentium 166 MHz/ 32 Mb/ 2.3 Gb, com sistema operacional MS-DOS e plataforma Microsoft Windows 95. Utilizou-se a ferramenta Borland Delphi 3.0 e a linguagem inerente Object Pascal, a qual permitiu uma codificação sob o paradigma da programação orientada para objetos.

5.8. O Ajuste dos Parâmetros dos Algoritmos Genéticos

Os parâmetros principais a serem ajustados em um algoritmo genético são o tamanho da população, a taxa de cruzamento e a taxa de mutação. Em decorrência da hibridização relacionada ao processo de seleção baseado no *Simulated Annealing*, o método proposto incorpora ainda a temperatura inicial e a taxa de resfriamento. Não existem regras ou relações lineares ou não-lineares que determinem a calibração destes parâmetros, e este é um processo que deve ser feito à partir de uma busca exaustiva. Porém, esta busca não faz parte dos objetivos deste trabalho, e portanto são adotados valores recomendados na literatura e/ou testados pelo autor com pequena amostragem. Admite-se com esta decisão que a robustez inerente aos Algoritmos Genéticos permita realizar os testes sem o ajuste fino.

De acordo com trabalhos como os de Grefenstette (1986), Goldberg (1989), Davis (1989), Johnson *et al.* (1989), Eglese (1990), Mitchell (1996) Reeves (1993) as seguintes faixas de valores são recomendáveis:

- Tamanho da População: 30 a 200;
  n a 2n, onde n = tamanho do cromossoma;
- Taxa de Cruzamento: 0,5 a 1,0;
- Taxa de Mutação: 0,001 a 0,005;
- Temperatura Inicial : $Te > 0$;
- Taxa de Resfriamento: $0 < Tr < 1$;
  0,8 a 0,99;

Adotou-se o seguinte conjunto de valores dos parâmetros para todos os problemas: tamanho da população igual a 40, o qual utilizado com 250 gerações, resulta em 10.000 operações genéticas; taxa de cruzamento igual a 0,9; taxa de decisão igual a 0,8; taxa de mutação igual a 0,4; temperatura inicial igual a 90; e taxa de resfriamento igual a 0,96. Com estes valores, pode-se verificar a opção pelo tamanho pequeno de população em troca do número maior de gerações; e pode-se também verificar que a taxa de Mutação de Modo é efetivamente superior ao estabelecido pela literatura, devido ao fato de o número de modos ser muito elevado nos problemas modelados.

5.9. Resultados dos Testes Realizados

A tabela 5.4 mostra os resultados dos trinta experimentos para os problema-teste. A primeira coluna contém os valores dos melhores resultados da função

objetivo, os quais expressam a razão entre a somatória dos valores absolutos dos desvios mensais dos recursos monetários e o somatório dos recursos disponíveis, em termos percentuais; a segunda coluna contém os tempos médios de processamento computacional; a terceira coluna mostra o número de experimentos que apresentou o melhor resultado de função objetivo; a quarta coluna utiliza a média aritmética μ como medida de posição; e a última coluna usa o desvio-padrão amostral σ como medida de dispersão dos dados ao entorno da tendência central.

$$\mu = (\sum_{Ne} X) / Ne \qquad [4]$$

$$\sigma = ((\sum_{Ne} (X - \mu)^2) / (Ne - 1)) ½ \qquad [5]$$

Onde X = Resultado de um Elemento do Espaço Amostral;
Ne = Número de Amostras;

Número Problema	Melhor Resultado	Tempo (minutos)	Número de Vezes	Média dos Resultados	Desvio Padrão dos Resultados
1	0,007802	01:03	2	0,020187	0,009084
2	0,001839	00:42	2	0,004668	0,002198
3	0,000266	00:29	2	0,005961	0,009107
4	0,000000	00:26	4	0,002077	0,001929
5	0,000052	00:26	1	0,004392	0,002713
6	0,000000	00:05	6	0,003816	0,000324
7	0,000000	00:04	40	0,000000	0,000000
8	0,000000	00:04	4	0,002297	0,011554

Tab. 5.4 - Resultado dos Experimentos Realizados usando os problemas-teste.

5.10. Gráficos dos Experimentos Realizados

Com base nos valores obtidos das tabelas anteriores, pode-se construir gráficos (figuras 5.1 a 5.6) que auxiliam na compreensão do desempenho do método proposto, à partir das variáveis número de atividades, CNC, T-*density*, função objetivo e tempo de processamento (em segundos). Para esta finalidade, foram utilizados somente os quatro projetos originais (problemas-teste 1 a 4). Desta análise, em que os seis gráficos comportam-se de maneira semelhante, pode-se concluir que a efetividade e a eficiência do método decrescem exponencialmente com o aumento do número de atividades e da complexidade do problema. A tabela 5.4 contém dados que respaldam esta conclusão, ao apresentar a maior freqüência de valores ótimos nos problemas 6 a 8.

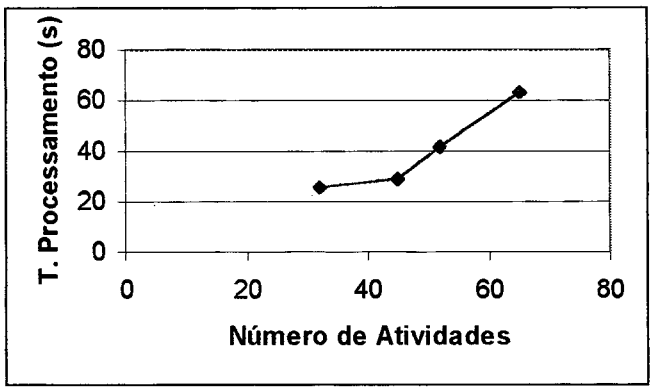


Figura 5.1 – Gráfico Número de Atividades x Tempo de Processamento Computacional: problemas-teste 1 a 4.

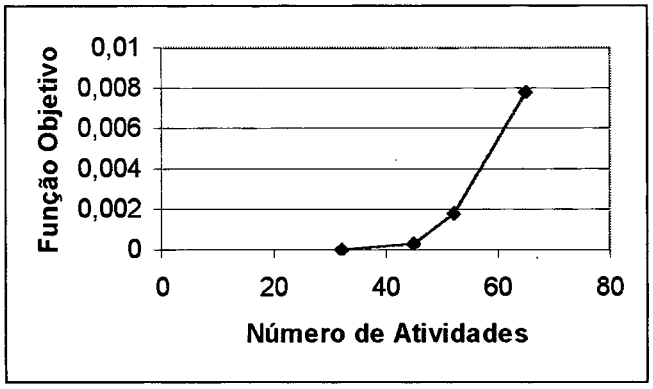


Figura 5.2 – Gráfico Número de Atividades x Função Objetivo: problemas-teste 1 a 4.

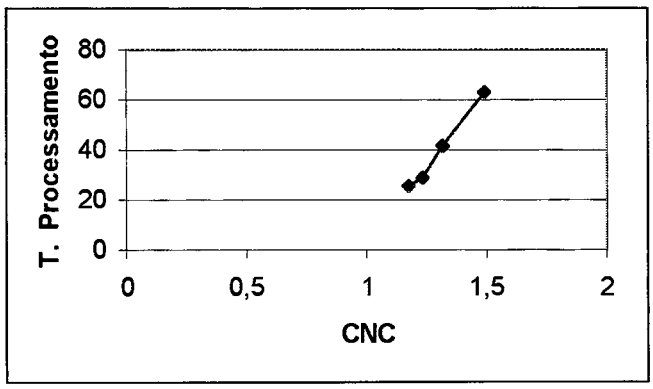


Figura 5.3 – Gráfico Complexidade CNC x Tempo de Processamento Computacional: problemas-teste 1 a 4.

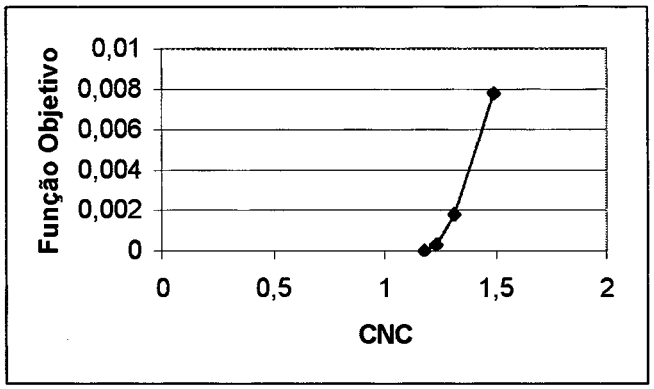


Figura 5.4 – Gráfico Complexidade CNC x Função Objetivo: problemas-teste 1 a 4.

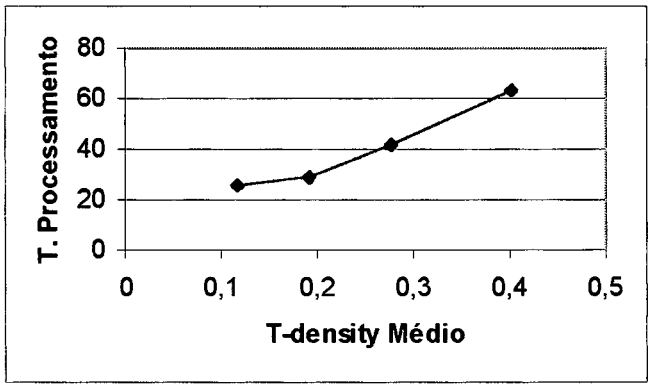


Figura 5.5 – Gráfico Complexidade T-*density* Médio x Tempo de Processsamento Computacional: problemas-teste 1 a 4.

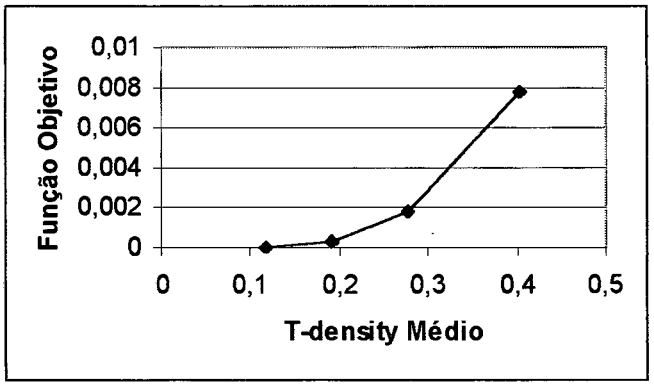


Figura 5.6 – Gráfico Complexidade T-*density* Médio x Função Objetivo: problemas-teste 1 a 4.

5.11. Performance do Algoritmo Genético

As figuras mostradas a seguir (5.7 a 5.22), contém os gráficos que expressam a performance do Algoritmo Genético em termos da evolução do valor da função objetivo, e em termos da evolução do valor médio da função objetivo para cada população de cromossomas, ambos em função do número de iterações. Cada dupla de gráficos diz respeito a um problema-teste, e consiste no melhor experimento de cada série de trinta, cujos resultados finais encontram-se na tabela 5.4, colunas 2 e 5.

Ao possibilitar a comparação entre o melhor resultado pertencente à última iteração (250) e o melhor resultado da população inicial, os gráficos evidenciam que o algoritmo melhora substancialmente o valor da função objetivo. Além deste indicativo, os gráficos permitem visualizar mais claramente que a população dos cromossomas evolui como um todo, conforme pode ser observado nos gráficos do valor médio da função objetivo para cada população ligada à cada iteração. O gráfico da função objetivo decresce em patamares devido à reprodução com elitismo, a qual replica os melhores cromossomas de uma população atual para a seguinte.

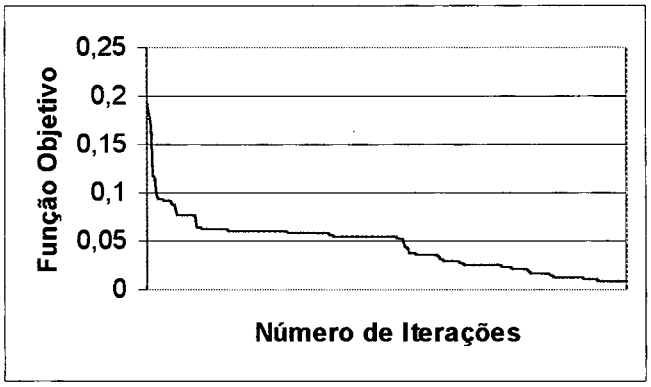


Fig. 5.7 – Evolução do Melhor Valor da Função Objetivo no Problema 1.

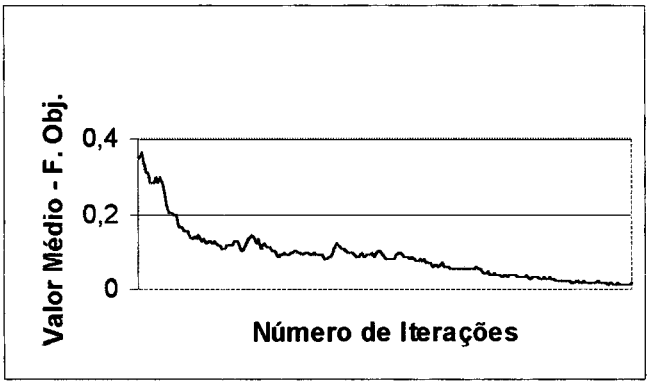


Fig. 5.8 – Evolução do Valor Médio da Função Objetivo no Problema 1.

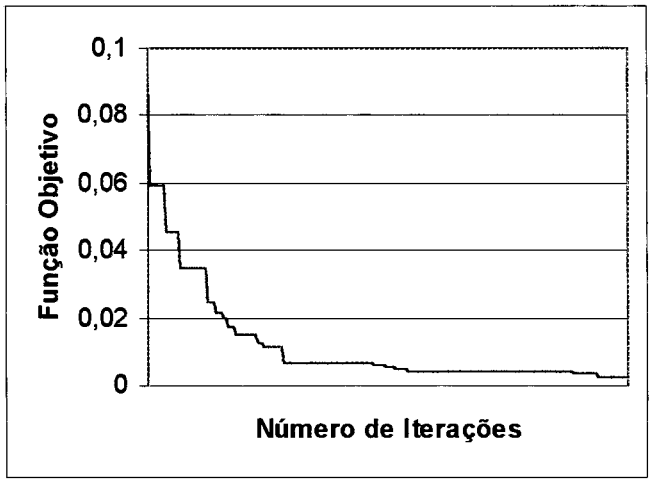


Fig. 5.9 – Evolução do Valor da Função Objetivo no Problema 2.

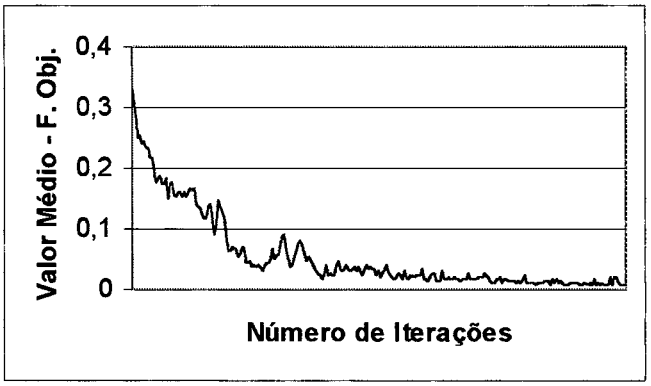


Fig. 5.10 – Evolução do Valor Médio da Função Objetivo no Problema 2.

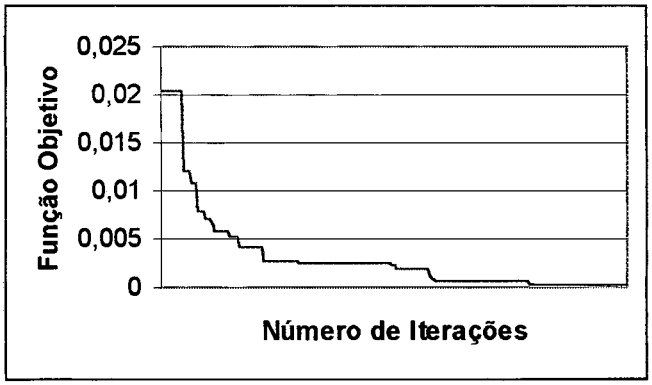


Fig. 5.11 – Evolução do Melhor Valor da Função Objetivo no Problema 3.

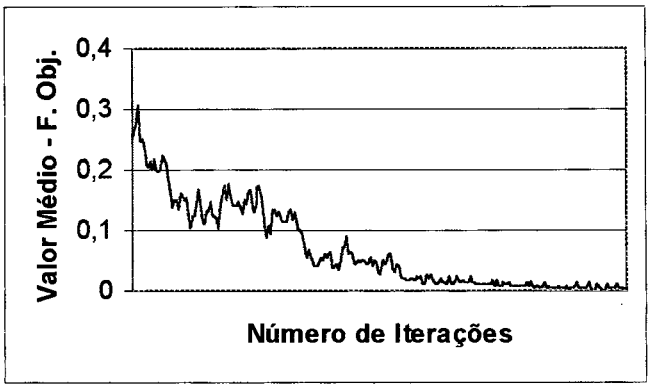


Fig. 5.12 – Evolução do Valor Médio da Função Objetivo no Problema 3.

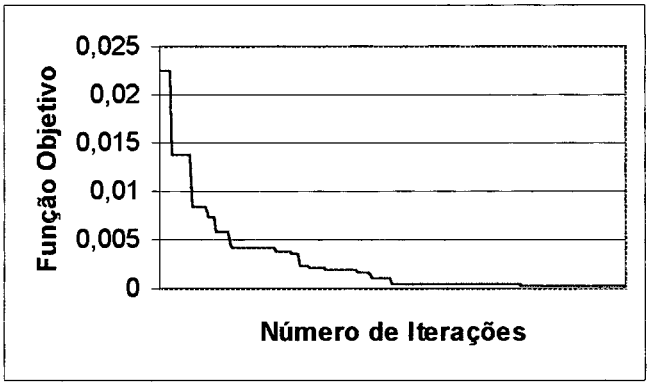


Fig. 5.13 – Evolução do Melhor Valor da Função Objetivo no Problema 4.

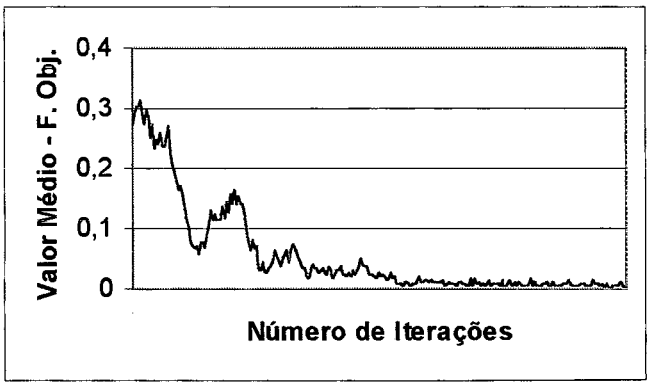


Fig. 5.14 – Evolução do Valor Médio da Função Objetivo no Problema 4.

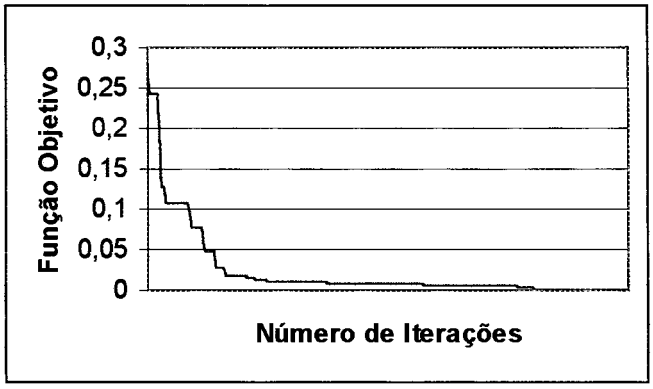


Fig. 5.15 – Evolução do Melhor Valor da Função Objetivo no Problema 5.

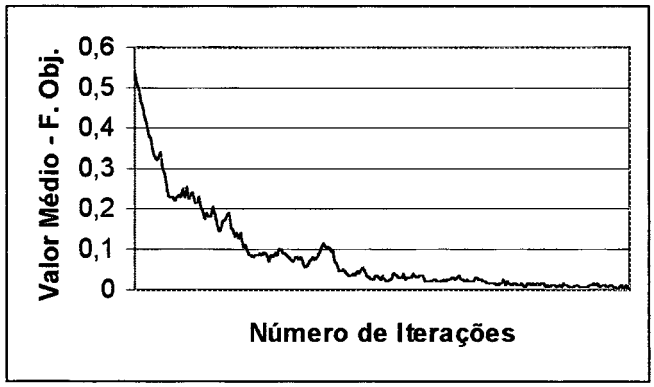


Fig. 5.16 – Evolução do Valor Médio da Função Objetivo no Problema 5.

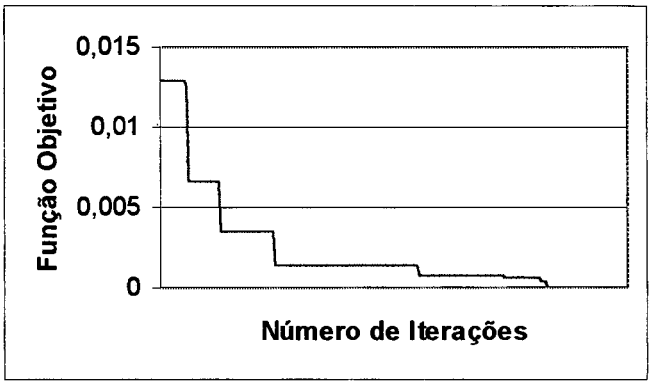


Fig. 5.17 – Evolução do Melhor Valor da Função Objetivo no Problema 6.

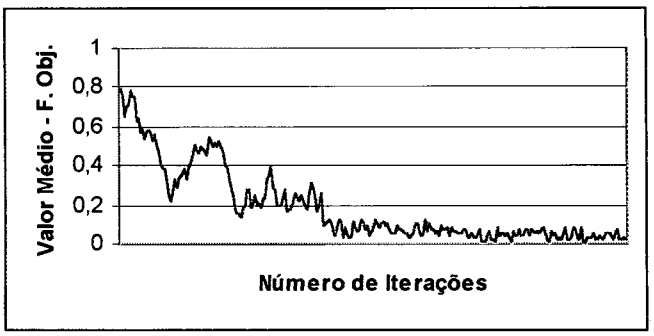


Fig. 5.18 – Evolução do Valor Médio da Função Objetivo no Problema 6.

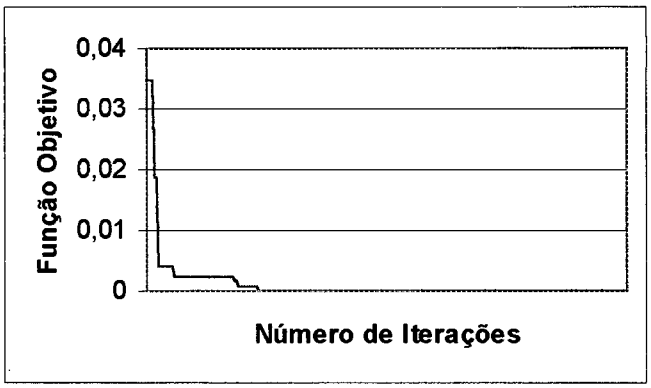


Fig. 5.19 – Evolução do Melhor Valor da Função Objetivo no Problema 7.

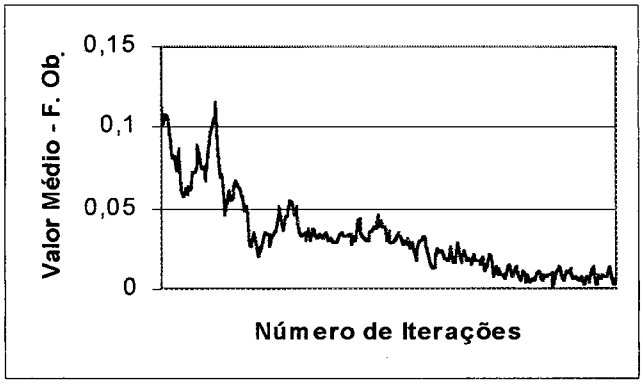


Fig. 5.20 – Evolução do Valor Médio da Função Objetivo no Problema 7.

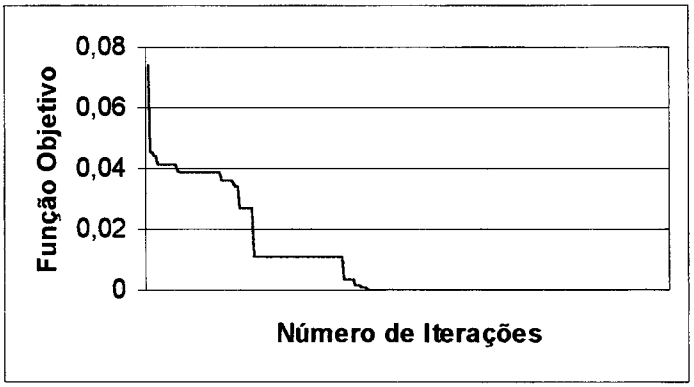


Fig. 5.21 – Evolução do Melhor Valor da Função Objetivo no Problema 8.

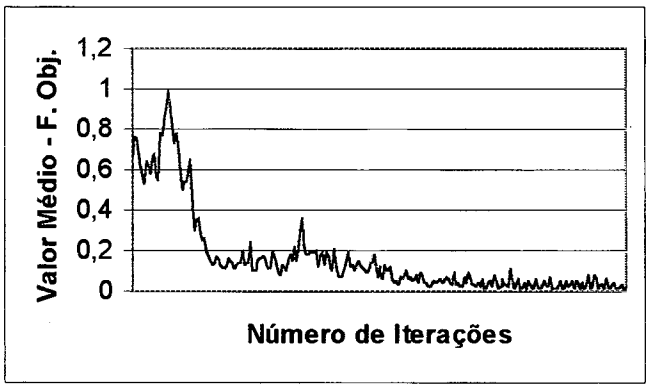


Fig. 5.22 – Evolução do Valor Médio da Função Objetivo no Problema 8.

5.12. Exemplos de Programas Gerados

Foram selecionados os problemas 3 e 5 para exemplificar a qualidade do atendimento ao critério de programação, bem como para demonstrar a flexibilidade do método proposto em relação à variações nas condições de contorno. A diferença entre os dois problemas reside no horizonte de programação (14 meses para o problema 3 e 12 meses para o problema 5) e no perfil da disponibilidade mensal de recursos monetários. Assim, o problema 5 consiste em uma compressão de projeto baseada no tempo, de dois meses, mantendo constante o somatório dos recursos monetários disponíveis.

As tabelas 5.5 a 5.12 referem-se ao problema 3: na 5.5 pode-se visualizar os valores dos recursos monetários mensais disponíveis; na 5.6 encontra-se o programa considerado ótimo e portanto àquele que possui desvio zero de recursos monetários; nas tabelas 5.7 e 5.8 pode-se ver os resultados da melhor programação obtida com o protótipo do método proposto, e portanto àquele cujo valor de função objetivo (0,000266) encontra-se na tabela 5.4; e nas tabelas 5.9 até 5.12 encontram-se outras duas programações com valores de função objetivo igual a 0,000655.

As tabelas 5.13 até 5.18 pertencem ao problema 5: nas tabelas 5.13 e 5.14 observa-se o programa que possui o melhor resultado de função objetivo (0,000052) dentre os trinta experimentos; nas tabelas seguintes encontram-se dois outros programas, com função objetivo igual a 0,000081 e 0,000096, respectivamente.

A figura 5.23 expressa graficamente o perfil da disponibilidade de recursos monetários para o problema 3; as figuras 5.24 até 5.35 são representações gráficas das disponibilidades e demandas de recursos monetários, bem como dos desvios mensais entre estas grandezas. Estas figuras estão associadas às tabelas 5.7, 5.9, 5.11, 5.13, 5.15 e 5.17.

Os programas mostrados nestas tabelas e figuras constituem indicativos do desempenho do método proposto, o qual é capaz de apresentar desvios mensais de recursos monetários considerados desprezíveis em se tratando deste tipo de recurso e do contexto da programação de obras de construção civil.

Mês	Desp. Proj.
1	38603
2	36302,48
3	82369,1738
4	63387,5962
5	6527,125
6	11091,6744
7	72546,5006
8	130462,527
9	30812,7387
10	165697,624
11	148928,031
12	11461,8824
13	12133,0471
14	1515,92

Tab. 5.5- Recursos Monetários Disponíveis para o Problema 3.

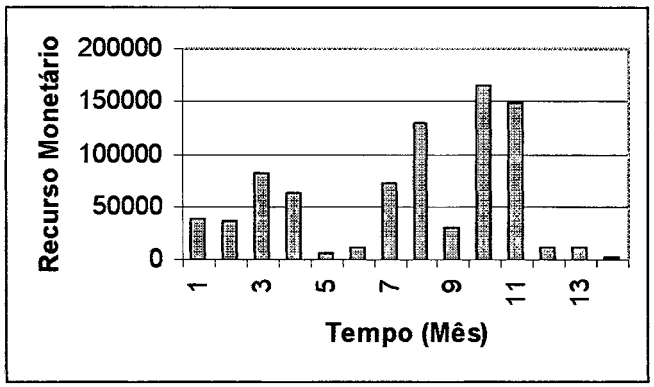


Fig. 5.23 - Histograma dos Recursos Monetários Disponíveis para o Problema 3.

Atividade	N Eq.	D	dTipo	Inicio	Final
0	0	0	0	0	0
1	8	5	0	1	5
2	2	3	0	6	8
3	1	2	0	9	10
4	10	13	0	11	23
5	10	12	0	11	22
6	20	1	0	24	24
7	5	2	0	25	26
8	12	25	5	27	51
9	9	10	2	44	53
10	13	11	2,2	46	56
11	5	5	1	53	57
12	14	48	9,6	75	122
13	13	26	0	58	83
14	1	5	1	123	127
15	1	7	1,4	123	129
16	6	15	3	123	137
17	1	11	2,2	123	133
18	1	6	1,2	123	128
19	1	3	0,6	123	125
20	2	8	1,6	130	137
21	5	5	1	126	130
22	12	9	1,8	131	139
23	6	10	2	126	135
24	1	7	1,4	125	131
25	1	16	3,2	130	145
26	6	14	2,8	135	148
27	12	16	3,2	138	153
28	6	11	2,2	141	151
29	12	40	8	142	181
30	6	26	5,2	144	169
31	6	22	4,4	138	159
32	5	23	4,6	152	174
33	11	48	9,6	157	204
34	6	26	5,2	185	210
35	4	26	5,2	191	216
36	9	25	5	192	216
37	12	24	4,8	198	221
38	6	9	1,8	215	223
39	1	8	1,6	216	223
40	6	14	2,8	211	224
41	18	51	10,2	218	268
42	2	19	3,8	145	163
43	2	16	3,2	214	229
44	6	13	2,6	259	271
45	5	25	5	265	286

Tab. 5.6 – Programa que conduz ao resultado considerado ótimo para o problema 3. Função Objetivo Igual a Zero.

Mês	Rec.Disp.	Desp. Proj.	Desvio
1	38603	38603	0
2	36302,48	36302,48	0
3	82369,17385	82369,17385	-1,5384E-07
4	63387,59615	63387,59615	1,53843E-07
5	6527,125	6527,125	0
6	11091,67435	11065,80682	25,86753282
7	72546,50057	72545,85127	0,649296947
8	130462,5265	130340,6741	121,8523834
9	30812,73868	30836,45269	-23,7140106
10	165697,6244	165759,8864	-62,2620834
11	148928,0311	148990,4242	-62,3931183
12	11461,88235	11461,88235	5,88261E-08
13	12133,04706	12133,04706	1,76473E-07
14	1515,92	1515,92	0

Tab. 5.7 – Melhor Resultado de Recursos Monetários para o Problema 3. Função Objetivo Igual a 0,0000266 em uma Série de 30 Experimentos.

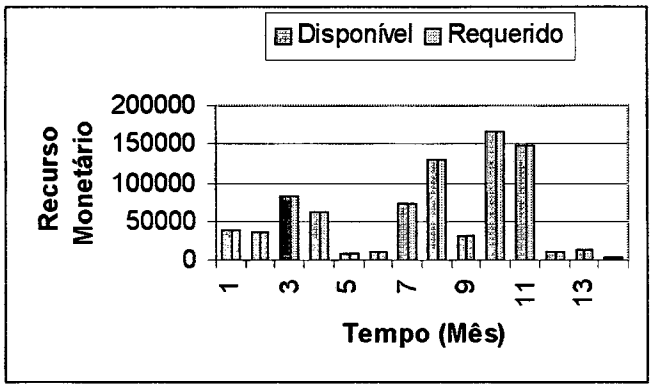


Fig. 5.24 – Perfil dos Recursos Monetários para o Melhor Experimento com o Problema 3.

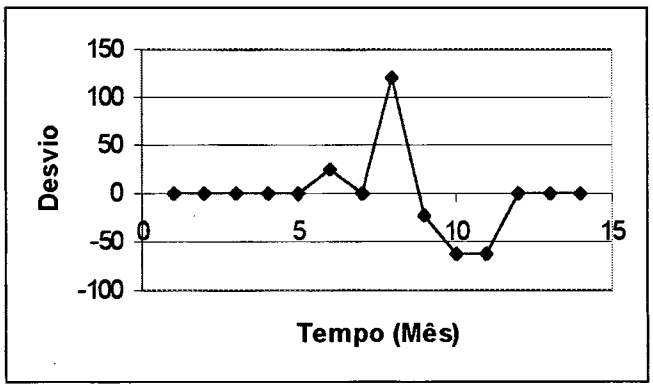


Fig. 5.25 – Gráfico dos Desvios Entre os Recursos Monetários Mensais Disponíveis e Requeridos para o Melhor Experimento com o Problema 3.

Atividade	N.Eq.	D	DTipo	Início	Final
0	0	0	0	0	0
1	5	7	0	1	7
2	3	2	0	8	9
3	2	1	0	10	10
4	11	12	0	11	22
5	12	10	0	11	20
6	18	1	0	24	24
7	4	2	0	25	26
8	12	25	5	27	51
9	8	11	2,2	44	54
10	15	9	1,8	48	56
11	5	5	1	53	57
12	14	48	9,6	75	122
13	13	26	0	58	83
14	1	5	1	123	127
15	2	4	0,8	123	126
16	8	11	2,2	125	135
17	1	11	2,2	123	133
18	1	6	1,2	123	128
19	2	2	0,4	123	124
20	2	8	1,6	130	137
21	4	6	1,2	124	129
22	11	10	2	129	138
23	8	7	1,4	129	135
24	1	7	1,4	125	131
25	3	6	1,2	130	135
26	8	10	2	132	141
27	15	13	2,6	134	146
28	7	9	1,8	135	143
29	15	32	6,4	163	194
30	10	16	3,2	137	152
31	8	17	3,4	148	164
32	7	17	3,4	152	168
33	14	38	7,6	156	193
34	9	17	3,4	187	203
35	5	21	4,2	194	214
36	11	21	4,2	194	214
37	14	21	4,2	199	219
38	9	6	1,2	216	221
39	1	8	1,6	216	223
40	6	14	2,8	209	222
41	18	51	10,2	218	268
42	2	19	3,8	211	229
43	2	16	3,2	217	232
44	6	13	2,6	259	271
45	5	25	5	265	286
46	0	0	0	0	0

Tab. 5.8 – Melhor Programa referente ao Problema 3, com Valor de Função Objetivo Igual a 0,0000266.

Mês	Rec.Disp.	Desp. Proj.	Desvio
1	38603	38603	0
2	36302,48	36302,48	0
3	82369,1738	82369,17385	-1,53843E-07
4	63387,5962	63387,59615	1,53843E-07
5	6527,125	6527,125	0
6	11091,6744	11040,79643	50,87792243
7	72546,5006	72545,87756	0,623007717
8	130462,527	130420,9804	41,54617622
9	30812,7387	30798,84003	13,89865326
10	165697,624	165750,083	-52,45865558
11	148928,031	148982,5182	-54,48710328
12	11461,8824	11461,88235	5,88261E-08
13	12133,0471	12133,04706	1,76473E-07
14	1515,92	1515,92	0

Tab. 5.9 – 2° Melhor Resultado de Recursos Monetários para o Problema 3. Função Objetivo Igual a 0,000655 em uma Série de 30 Experimentos.

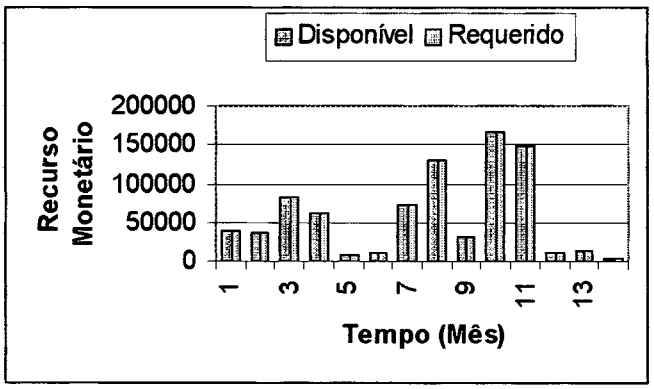


Fig. 5.26 – Perfil dos Recursos Monetários para o 2° Melhor Experimento com o Problema 3.

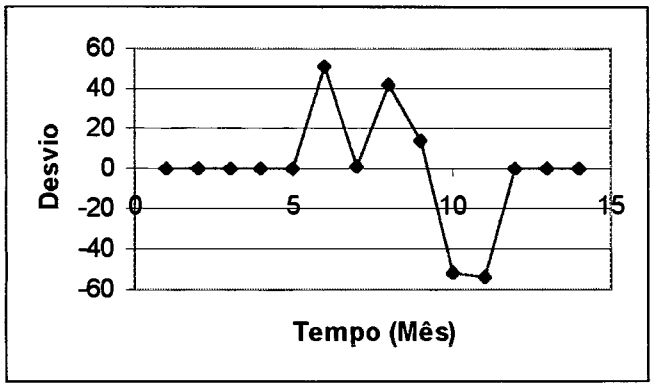


Fig. 5.27 – Gráfico dos Desvios Entre os Recursos Monetários Mensais Disponíveis e Requeridos para o 2° Melhor Experimento com o Problema 3.

Atividade	N.Eq.	d	DTipo	Inicio	Final
0	0	0	0	0	0
1	6	6	0	1	6
2	3	2	0	7	8
3	1	2	0	9	10
4	12	11	0	11	21
5	9	13	0	11	23
6	17	1	0	24	24
7	5	2	0	25	26
8	12	25	5	27	51
9	9	10	2	44	53
10	14	10	2	46	55
11	6	4	0,8	53	56
12	14	48	9,6	75	122
13	13	26	0	58	83
14	2	3	0,6	123	125
15	1	7	1,4	124	130
16	9	10	2	123	132
17	2	6	1,2	123	128
18	2	3	0,6	123	125
19	3	1	0,2	129	129
20	2	8	1,6	130	137
21	3	8	1,6	129	136
22	11	10	2	129	138
23	6	10	2	125	134
24	2	4	0,8	124	127
25	4	4	0,8	130	133
26	8	10	2	132	141
27	13	15	3	146	160
28	8	8	1,6	136	143
29	13	37	7,4	172	208
30	8	20	4	138	157
31	7	19	3,8	134	152
32	8	15	3	152	166
33	15	35	7	157	191
34	8	20	4	184	203
35	7	15	3	193	207
36	12	19	3,8	194	212
37	15	19	3,8	198	216
38	8	7	1,4	216	222
39	3	3	0,6	216	218
40	7	12	2,4	213	224
41	18	51	10,2	218	268
42	3	13	2,6	143	155
43	5	7	1,4	220	226
44	6	13	2,6	259	271
45	5	25	5	265	286
46	0	0	0	0	0

Tab. 5.10 – 2° Melhor Programa referente ao Problema 3, com Valor de Função Objetivo Igual a 0,000655.

Mês	Rec.Disp	Desp. Proj.	Desvio
1	38603	38603	0
2	36302,48	36302,48	0
3	82369,1738	82369,1738	-1,5384E-07
4	63387,5962	63387,5962	1,5384E-07
5	6527,125	6527,125	0
6	11091,6744	11040,7964	50,8779224
7	72546,5006	72545,8776	0,62300772
8	130462,527	130420,98	41,5461762
9	30812,7387	30798,84	13,8986533
10	165697,624	165750,083	-52,4586556
11	148928,031	148982,518	-54,4871033
12	11461,8824	11461,8824	5,8826E-08
13	12133,0471	12133,0471	1,7647E-07
14	1515,92	1515,92	0

Tab. 5.11 – 3° Melhor Resultado de Recursos Monetários para o Problema 3. Função Objetivo Igual a 0,000655 em uma Série de 30 Experimentos.

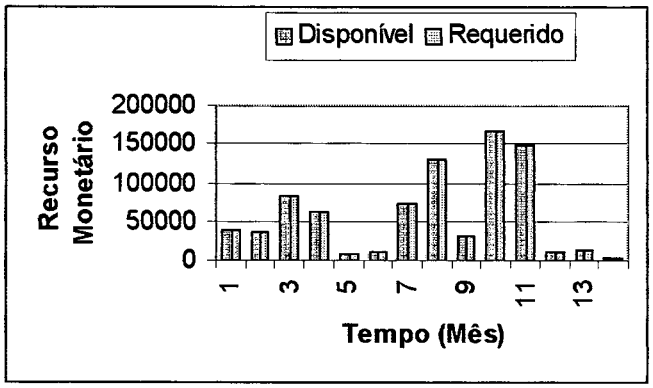


Fig. 5.28 – Perfil dos Recursos Monetários para o 3° Melhor Experimento com o Problema 3.

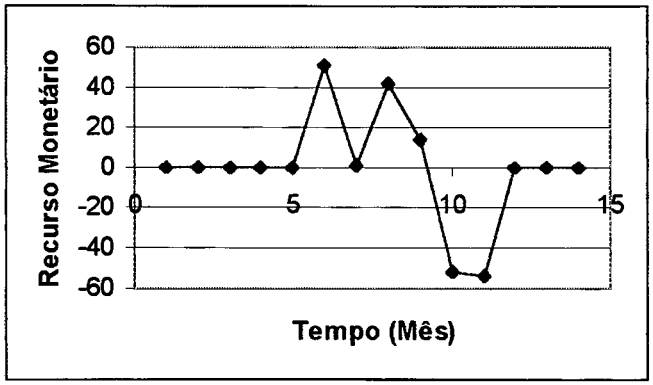


Fig. 5.29 – Gráfico dos Desvios Entre os Recursos Monetários Mensais Disponíveis e Requeridos para o 3° Melhor Experimento com o Problema 3.

Atividade	N.Eq.	d	DTipo	Início	Final
0	0	0	0	0	0
1	6	6	0	1	6
2	3	2	0	7	8
3	1	2	0	9	10
4	12	11	0	11	21
5	9	13	0	11	23
6	17	1	0	24	24
7	5	2	0	25	26
8	12	25	5	27	51
9	9	10	2	44	53
10	14	10	2	46	55
11	6	4	0,8	53	56
12	14	48	9,6	75	122
13	13	26	0	58	83
14	2	3	0,6	123	125
15	1	7	1,4	124	130
16	9	10	2	123	132
17	2	6	1,2	123	128
18	2	3	0,6	123	125
19	3	1	0,2	129	129
20	2	8	1,6	130	137
21	3	8	1,6	129	136
22	11	10	2	129	138
23	6	10	2	125	134
24	2	4	0,8	124	127
25	4	4	0,8	130	133
26	8	10	2	132	141
27	13	15	3	146	160
28	8	8	1,6	136	143
29	13	37	7,4	172	208
30	8	20	4	138	157
31	7	19	3,8	134	152
32	8	15	3	152	166
33	15	35	7	157	191
34	8	20	4	184	203
35	7	15	3	193	207
36	12	19	3,8	194	212
37	15	19	3,8	198	216
38	8	7	1,4	216	222
39	3	3	0,6	216	218
40	7	12	2,4	213	224
41	18	51	10,2	218	268
42	3	13	2,6	143	155
43	5	7	1,4	220	226
44	6	13	2,6	259	271
45	5	25	5	265	286
46	0	0	0	0	0

Tab. 5.12 – 3° Melhor Programa referente ao Problema 3, com Valor de Função Objetivo Igual a 0,000655.

Mês	Rec.Disp.	Desp. Proj.	Desvio
1	37163,53846	37163,53846	4,61536E-07
2	38578,40225	38578,40225	7,60119E-08
3	81532,71314	81532,71313	3,08602E-07
4	63624,94615	63624,94615	1,53843E-07
5	6962,266667	6962,266667	3,33333E-07
6	19327,60886	19306,60886	21,00000019
7	129941,8819	129958,7627	-16,88080484
8	85387,35602	85391,47521	-4,11919532
9	110709,8223	110709,8223	3,48431E-07
10	214469,0153	214469,0153	-1,22469E-07
11	10939,48852	10939,48852	-4,08161E-07
12	12949,62713	12949,62713	1,49661E-07

Tab. 5.13 – Melhor Resultado de Recursos Monetários para o Problema 5. Função Objetivo Igual a 0,000052 em uma Série de 30 Experimentos.

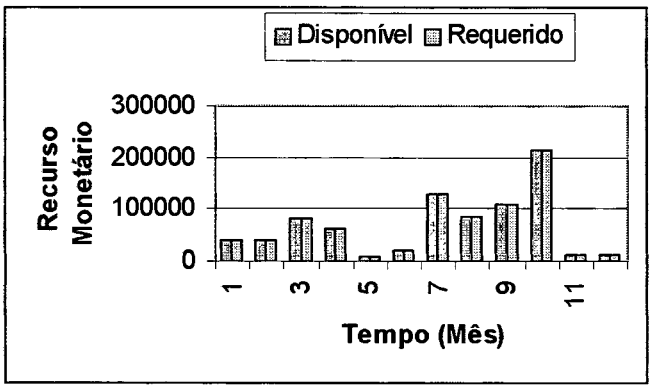


Fig. 5.30 – Perfil dos Recursos Monetários para o Melhor Experimento com o Problema 5.

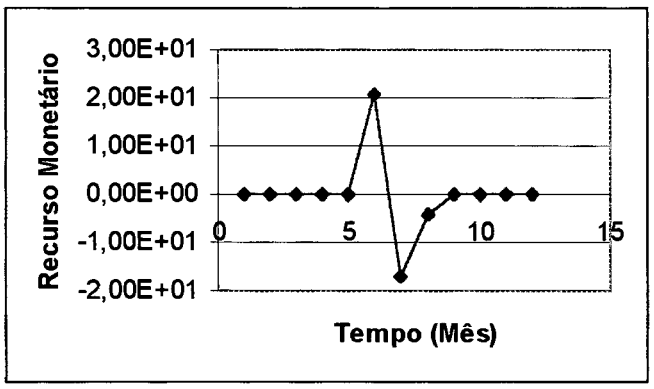


Fig. 5.31 – Gráfico dos Desvios Entre os Recursos Monetários Mensais Disponíveis e Requeridos para o Melhor Experimento com o Problema 5.

Atividade	N.Eq	d	dTipo	Inicio	Final
0	0	0	0	0	0
1	9	4	0	1	4
2	3	2	0	5	6
3	2	1	0	7	7
4	10	13	0	12	24
5	10	12	0	8	19
6	19	1	0	25	25
7	5	2	0	26	27
8	13	23	4,6	28	50
9	8	11	2,2	43	53
10	12	11	2,2	46	56
11	5	5	1	53	57
12	15	45	9	75	119
13	13	26	0	58	83
14	3	2	0,4	120	121
15	1	7	1,4	120	126
16	7	13	2,6	120	132
17	3	4	0,8	120	123
18	2	3	0,6	120	122
19	1	3	0,6	120	122
20	3	6	1,2	130	135
21	5	5	1	123	127
22	10	11	2,2	125	135
23	6	10	2	121	130
24	2	4	0,8	126	129
25	2	8	1,6	130	137
26	7	12	2,4	132	143
27	14	14	2,8	135	148
28	5	13	2,6	135	147
29	14	34	6,8	138	171
30	8	20	4	138	157
31	8	17	3,4	135	151
32	6	19	3,8	143	161
33	13	41	8,2	147	187
34	8	20	4	172	191
35	5	21	4,2	176	196
36	11	21	4,2	176	196
37	14	21	4,2	181	201
38	7	8	1,6	196	203
39	1	8	1,6	196	203
40	7	12	2,4	193	204
41	19	49	9,8	198	246
42	2	19	3,8	147	165
43	2	16	3,2	196	211
44	5	16	3,2	235	250
45	7	18	3,6	242	256
46	0	0	0	0	0

Tab. 5.14 – Melhor Programa referente ao Problema 5, com Valor de Função Objetivo Igual a 0,000052.

Mês	Rec.Disp.	Desp. Proj.	Desvio
1	37163,53846	37163,53846	4,61536E-07
2	38578,40225	38578,40225	7,60119E-08
3	81532,71314	81532,71313	3,08602E-07
4	63624,94615	63624,94615	1,53843E-07
5	6962,266667	6962,266667	3,33333E-07
6	19327,60886	19294,86795	32,74090928
7	129941,8819	129959,4649	-17,58301424
8	85387,35602	85402,51391	-15,15789501
9	110709,8223	110709,8223	3,48431E-07
10	214469,0153	214469,0153	-1,22469E-07
11	10939,48852	10939,48852	-4,08161E-07
12	12949,62713	12949,62713	1,49661E-07

Tab. 5.15 – 2° Melhor Resultado de Recursos Monetários para o Problema 5. Função Objetivo Igual a 0,000081 em uma Série de 30 Experimentos.

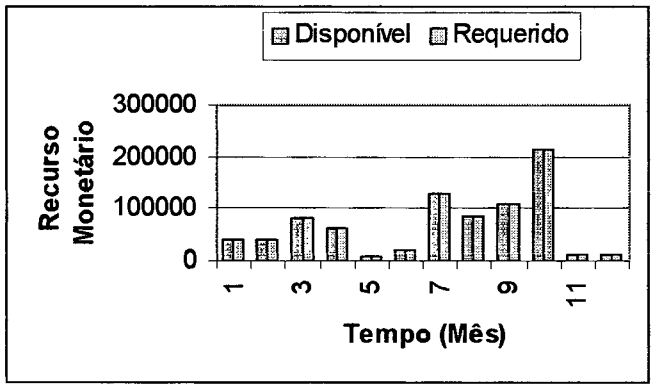


Fig. 5.32 – Perfil dos Recursos Monetários para o 2° Melhor Experimento com o Problema 5.

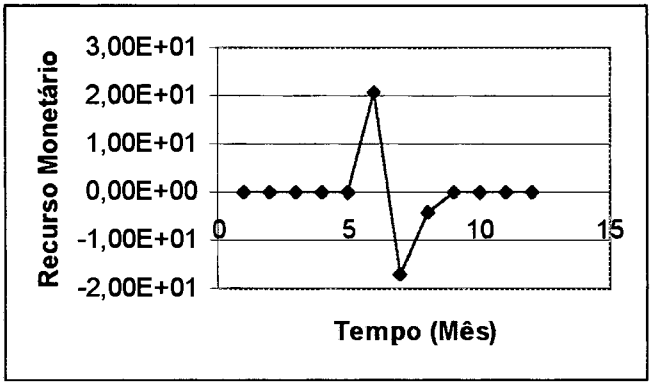


Fig. 5.33 – Gráfico dos Desvios Entre os Recursos Monetários Mensais Disponíveis e Requeridos para o 2° Melhor Experimento com o Problema 5.

Atividade	N.Eq.	d	DTipo	Inicio	Final
0	0	0	0	0	0
1	9	4	0	1	4
2	2	3	0	5	7
3	2	1	0	8	8
4	10	13	0	12	24
5	10	12	0	9	20
6	19	1	0	25	25
7	4	2	0	26	27
8	13	23	4,6	28	50
9	8	11	2,2	43	53
10	13	11	2,2	46	56
11	5	5	1	53	57
12	15	45	9	75	119
13	13	26	0	58	83
14	2	3	0,6	120	122
15	2	4	0,8	120	123
16	7	13	2,6	120	132
17	2	6	1,2	120	125
18	1	6	1,2	120	125
19	2	2	0,4	120	121
20	3	6	1,2	130	135
21	4	6	1,2	125	130
22	11	10	2	125	134
23	7	8	1,6	122	129
24	1	7	1,4	122	128
25	2	8	1,6	130	137
26	7	12	2,4	132	143
27	12	16	3,2	135	150
28	5	13	2,6	135	147
29	14	34	6,8	139	172
30	8	20	4	138	157
31	8	17	3,4	135	151
32	6	19	3,8	143	161
33	13	41	8,2	147	187
34	8	20	4	172	191
35	5	21	4,2	176	196
36	11	21	4,2	176	196
37	14	21	4,2	181	201
38	6	9	1,8	195	203
39	1	8	1,6	196	203
40	7	12	2,4	193	204
41	19	49	9,8	198	246
42	2	19	3,8	145	163
43	2	16	3,2	196	211
44	5	16	3,2	235	250
45	7	18	3,6	242	256
46	0	0	0	0	0

Tab. 5.16 – 2° Melhor Programa referente ao Problema 5, com Valor de Função Objetivo Igual a 0,000081.

Mês	Rec.Disp.	Desp. Proj.	Desvio
1	37163,53846	37163,53846	4,61536E-07
2	38578,40225	38578,40225	7,60119E-08
3	81532,71314	81532,71313	3,08602E-07
4	63624,94615	63624,94615	1,53843E-07
5	6962,266667	6962,266667	3,33333E-07
6	19327,60886	19288,6634	38,94545474
7	129941,8819	129980,8273	-38,94545443
8	85387,35602	85387,35602	-2,73794E-07
9	110709,8223	110709,8223	3,48431E-07
10	214469,0153	214469,0153	-1,22469E-07
11	10939,48852	10939,48852	-4,08161E-07
12	12949,62713	12949,62713	1,49661E-07

Tab. 5.17 – 3° Melhor Resultado de Recursos Monetários para o Problema 5. Função Objetivo Igual a 0,000096 em uma Série de 30 Experimentos.

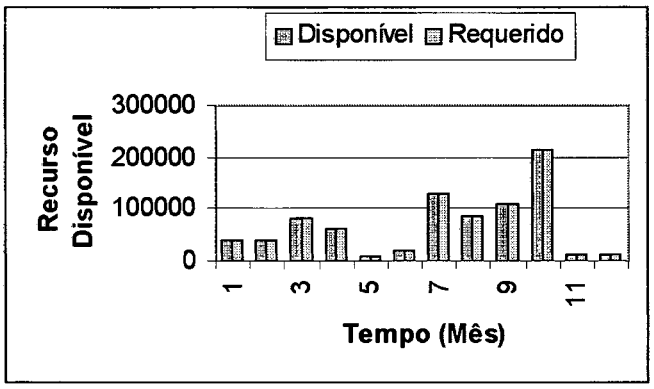


Fig. 5.34 – Perfil dos Recursos Monetários para o 3° Melhor Experimento com o Problema 5.

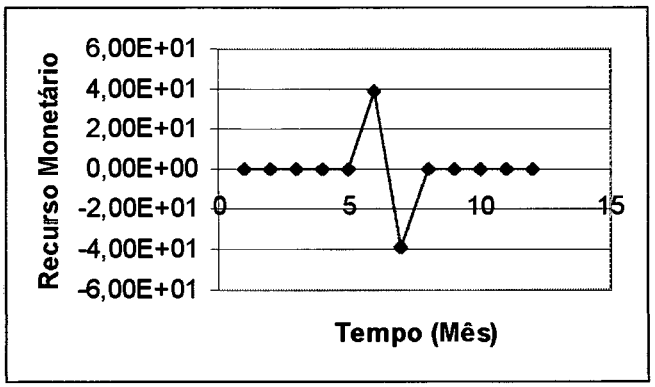


Fig. 5.35 – Gráfico dos Desvios Entre os Recursos Monetários Mensais Disponíveis e Requeridos para o 3° Melhor Experimento com o Problema 5.

Atividade	N.Eq.	d	DTipo	Inicio	Final
0	0	0	0	0	0
1	9	4	0	1	4
2	3	2	0	5	6
3	2	1	0	7	7
4	10	13	0	12	24
5	11	11	0	8	18
6	18	1	0	25	25
7	4	2	0	26	27
8	13	23	4,6	28	50
9	8	11	2,2	43	53
10	12	11	2,2	46	56
11	5	5	1	53	57
12	15	45	9	75	119
13	13	26	0	58	83
14	1	5	1	120	124
15	2	4	0,8	120	123
16	7	13	2,6	120	132
17	1	11	2,2	120	130
18	3	2	0,4	120	121
19	2	2	0,4	120	121
20	3	6	1,2	130	135
21	5	5	1	121	125
22	11	10	2	125	134
23	7	8	1,6	125	132
24	3	3	0,6	121	123
25	3	6	1,2	130	135
26	7	12	2,4	132	143
27	13	15	3	135	149
28	5	13	2,6	135	147
29	14	34	6,8	139	172
30	8	20	4	138	157
31	7	19	3,8	135	153
32	6	19	3,8	143	161
33	13	41	8,2	147	187
34	8	20	4	172	191
35	5	21	4,2	176	196
36	11	21	4,2	176	196
37	14	21	4,2	181	201
38	7	8	1,6	196	203
39	1	8	1,6	196	203
40	7	12	2,4	193	204
41	19	49	9,8	198	246
42	2	19	3,8	139	157
43	2	16	3,2	196	211
44	5	16	3,2	235	250
45	7	18	3,6	242	256
46	0	0	0	0	0

Tab. 5.18 – 3° Melhor Programa referente ao Problema 5, com Valor de Função Objetivo Igual a 0,000096.

CAPÍTULO VI

CONCLUSÕES E RECOMENDAÇÕES

6.1. Conclusões

Neste trabalho de tese foi apresentado um método de solução heurístico para a programação de edifícios dotados de múltiplos pavimentos-tipo, apropriado para a implementação computacional. A pesquisa está inserida no campo da Programação de Projetos com restrição de recursos e múltiplos modos, o qual é caracterizado por uma complexidade combinatorial de tal significância, que permite a solução exata apenas aos problemas teóricos de tamanho pequeno.

O modelo concebido evidencia a capacidade de poder ser utilizado futuramente em pacotes comerciais informatizados, devido à eficiência e à efetividade dos resultados, bem como pela capacidade de gerar interfaces amigáveis com o usuário final. Ao analisar sob o prisma da adequação às características reais da programação de obras, destaca-se o potencial de sob os princípios do método proposto, ser possível obter modelagens com relativo baixo nível de abstração; esta conclusão foi demonstrada pela aceitação de seis tipos importantes de restrições, pela assimilação de longos horizontes de tempo, e pela aceitação de diversas circunstâncias e disposições das atividades que compõem o tipo de projeto abordado.

A originalidade do trabalho pode ser descrita em três aspectos principais: apresentar um método de programação de projetos capaz de reunir importantes conhecimentos necessários ao tipo construtivo abordado (tais como a lógica horizontal, os *buffers*, as datas condicionantes, a lógica vertical e a restrição de espaço físico); ser capaz de apresentar resultados que podem variar do bom ao ótimo, ao utilizar os Algoritmos Genéticos; e evidenciar um caminho de integração entre as áreas de conhecimento envolvidas.

A adoção de um método de programação linear para modelar as atividades repetitivas, o Método da Linha de Balanço, resultou na mudança da orientação da programação pelo tempo para a orientação pelos recursos do projeto, bem como resultou na garantia da continuidade do trabalho das equipes alocadas, evitando o erro denominado *progresso fora de lógica*. O método proposto evidencia os principais tipos de recursos, o monetário e a mão-de-obra, para constituir-se em um auxílio realístico à tomada de decisão; os fluxos de recurso monetário são considerados periódicos (meses), mas o processo de programação das atividades e de alocação dos recursos funciona com unidades de tempo (dias), como em situações do mundo real.

A integração de uma Heurística de Programação Baseada em Regras de Prioridade com um algoritmo de busca evolutivo, foi interessante para a incorporação

simples e precisa do conhecimento específico relativo ao problema. Esta heurística, reconhecida na literatura como flexível e eficiente, teve sua regra de prioridade substituída pela escolha aleatória, o que possibilitou a geração de uma diversidade de programas. O esquema de geração de programas, que representa o segundo componente deste tipo de heurística, foi constituído pelo Esquema de Programação Serial, devido à compatibilidade com a estrutura de solução estabelecida no módulo de busca.

Na escolha de uma heurística apropriada para pesquisar o espaço de soluções, tornou-se interessante utilizar os Algoritmos Genéticos porque o espaço a ser pesquisado é grande, não é liso, não é unimodal, e o problema a ser solucionado não requer solução ótima. Esta decisão resultou na transferência das seguintes vantagens para o método proposto: a desenvoltura em espaços grandes de busca; o dispensar de suposições de linearidade, convexidade e diferencialidade; a capacidade de transpor limitações de otimalidade local; o Paralelismo Implícito, a propriedade de não melhorar apenas uma única solução, mas o conjunto; e a facilidade de modificação.

O algoritmo genético criado possui as seguintes características: sobre a estruturação dos Cromossomas, duas variáveis de decisão foram utilizadas, a priorização ou sequenciamento das atividades e os modos de produção; a posição relativa dos Genes no Cromossoma não corresponde ao código numérico das atividades; a codificação adotada é a de números inteiros representando as atividades e os modos; a adoção da reprodução com Elitismo permitiu que as melhores soluções fossem preservadas, evitando a destruição pelas operações de mutação e cruzamento, bem evitando a perda pela aleatoriedade do operador de Seleção; os operadores de seleção, implementam uma pressão de seleção progressiva, evitando a convergência prematura e promovendo alta diversidade nas populações iniciais; os operadores de cruzamento, inversão e mutação apresentados são orientados para o tipo de modelo em questão, em que a ordem dos genes é uma grandeza importante; e outro aspecto, é que estes operadores foram divididos em dois grupos, conforme o projeto seja modelado sobre uma rede linear ou sobre uma rede mista.

Para a elaboração do método proposto, houve a necessidade de integrar conhecimentos relativos a três áreas principais: a Construção Civil, através dos métodos de programação linear, particularmente o Método da Linha de Balanço; a Inteligência Artificial, através dos Algoritmos Genéticos; e a Pesquisa Operacional, através dos estudos dos problemas de programação da produção. A pesquisa sobre a literatura técnica mais antiga até os desenvolvimentos mais recentes destas áreas, possibilitaram a geração de uma síntese não trivial, a qual indica um caminho passível de ser continuado em pesquisas futuras.

O desenvolvimento desta pesquisa é relevante para as três áreas citadas: para a Construção Civil porque a programação de projetos é vital, e existe a demanda por ferramentas que apresentem bom desempenho; para a Inteligência Artificial e a Pesquisa Operacional, porque consiste em uma aplicação algorítmica híbrida dos Algoritmos Genéticos em um problema de produção real.

Na pesquisa bibliográfica presente nos diversos capítulos desta tese, são citados muitos trabalhos diretamente ligados ao tema da Programação de Projetos. O

autor desconhece trabalhos nacionais e internacionais, publicados em periódicos importantes ou presentes na Internet, que sejam destinados à programação de edifícios dotados de múltiplos pavimentos-tipo, os quais possuam o nível de abstração e as características em conjunto que caracterizam o método de solução proposto. Em resumo, estas características são:

- Pesquisar e integrar quatro ferramentas importantes na área da Programação de Projetos: (1) os Algoritmos Genéticos, (2) as Heurísticas de Programação Baseadas em Regras de Prioridade, (3) os Grafo Orientados, e (4) o Método da Linha de Balanço.

- Possuir duas vertentes: a direcionada para projetos modelados por redes lineares e a direcionada para redes mistas;

- Admitir múltiplos modos de execução para as atividades, onde cada modo represente o incremento de equipes na direção horizontal da construção, bem como na direção vertical;

- Promover a possibilidade de as atividades serem programadas com ritmos diferentes e naturais;

- Permitir integrações entre atividades repetitivas e não repetitivas, bem como entre atividades repetitivas com diferentes sentidos de execução (ascendente-ascendente, ascendente-descendente, descendente-descendente e descendente-ascendente);

- Conter uma solução orientada para a programação específica do tipo construtivo abordado, que programe o projeto como um todo, e que seja capaz de ser utilizado em pequenos, médios e grandes projetos (em termos de tempo, recursos, número de atividades e número de interrelacionamentos entre atividades) eficientemente e eficazmente;

- Conter a seguinte hibrização dos Algoritmos Genéticos: (1) representar os cromossomas com codificação não binária; (2) possuir duas variáveis de decisão para os cromossomas (sequenciamento e modo); (3) utilizar um método de seleção com pressão de seleção progressiva; (4) utilizar esquemas de recombinação orientados para problemas onde a ordem dos genes é importante; e (5) possuir incorporação de conhecimentos específicos do problema, através do processo de alocação de recursos e da população inicial;

Para proporcionar indicativos sobre o desempenho do método, foram realizados 240 experimentos utilizando 8 problemas fundamentados em projetos reais. Os testes indicam que o algoritmo evolui de maneira consistente, minimizando progressivamente o valor da função objetivo e melhorando a aptidão da população como um todo. Considerando que o objetivo de programação está ligado ao recurso monetário, e que portanto a precisão dos desníveis mensais não chega aos valores decimais, todos os experimentos foram considerados satisfatórios. Outro motivo que

respalda esta afirmação, é que os métodos exatos diminuem mais significativamente sua eficiência, à medida em que os problemas aumentam de tamanho e complexidade.

Os gráficos apresentados foram indicadores do comportamento exponencial da eficiência e da efetividade, em relação ao tamanho e à complexidade dos problemas. Também é importante ressaltar que os problemas testaram o desempenho do método proposto face a perfis de disponibilidade dos recursos monetários irregulares e de grande amplitude, assim como face à explosão combinatorial gerada pelo elevado número de modos de produção das atividades.

6.2. Recomendações

A competitividade entre as empresas ligadas à Construção Civil, a escassez e/ou o alto custo de insumos, assim como a complexidade dos problemas de programação são alguns dos motivos que induzem a pesquisa acerca das principais linhas de pesquisa que compuseram esta tese: o modelamento de projetos através de grafos e métodos de programação linear, como o método da Linha de Balanço; os problemas de programação de projetos, particularmente o Problema da Programação de Projetos com Restrição de Recursos (RCPSP); e os métodos de solução heurísticos, como os Algoritmos Genéticos.

Ao analisar o método proposto, pode-se deduzir de imediato que existe um campo aberto para o estudo da adoção de novos critérios de otimização, e de outras regras e restrições que o tornem mais específico à novos interesses. A calibração rigorosa dos parâmetros relativos aos Algoritmos Genéticos e à Seleção Boltzmann, também consiste em um importante trabalho futuro.

Ao rever a caracterização de um RCPSP descrita no capítulo II, pode-se recomendar outros desenvolvimentos: múltiplos objetivos de otimização; uso de informações ou dados probabilísticos; possibilidade de interrupção das atividades; abordagem de outros tipos de recursos; avaliação simultânea de um número maior de recursos; e a programação simultânea de múltiplos projetos.

O desenvolvimento de novas heurísticas, fundamentadas por exemplo no *Simulated Annealing* e no *Tabu Search*, constituem extensões importantes desta pesquisa, a fim de que sejam comparados formalmente os desempenhos. Pelo mesmo motivo, é interessante o estudo acerca de hibridizações entre métodos heurísticos.

Finalmente, o desenvolvimento de pacotes comerciais adequados para serem utilizados por empresas construtoras, associados à uma filosofia de produção, é outro campo vasto de pesquisa a ser promovido.

BIBLIOGRAFIA

AARTS, E., KORST, J. *Simulated Annealing and Boltzmann Machines.* Wiley, Nova York: 1989.

ABRAMSON, D., *Constructing School Timetables Using Simulated Annealing: Sequencial and Parallel Algorithms.* Management Science, 37, pág. 98-113: 1991.

ADELI, H., KARIM, A., scheduling/Cost Optimization and Neural Dynamics Model for Construction. Journal of Construction Engineering and Management, V. 123, 4, pág. 450-458: 1997.

AHUJA, H.N. *Project management.* John Wiley and Sons, Inc., New York: 1984.

AL SARRAJ, Z. M., *Formal Development of Line-of-Balance Technique. Journal of Construction Engineering and Management*, ASCE, v. 116, n. 4, p. 689-704: 1990.

ALVAREZ-VALDES, R., e TAMARIT, J. M., *Heuristic Algorithms for Resource - Constrained Project Scheduling: a Review and na Empirical Analisys*, in: R. Slowinski and J. Weglarz Ed., Advances in Project Scheduling, Elsevier, Amstrdam, p. 113-134: 1989.

ANDERSON, E. J. *Theory and Methodology: Mechanisms for Local Search.* European Journal of Operational Research. N° 88. P. 139-151: 1996.

ANDERSON, H., FJOSNE, A., SOLBERG, O. *A Network Model for Resource Allocation and Time Scheduling Specially Constructed for Repetitive Processes in the Building Industry*, Application of Critical Path Techniques, J. Brennan, The English Universities Press Ltd., London:1968.

ANTILL, J. M., WOODHEAD, R. W., *Critical Path Methods in Construction Practice*, Willey-Interscience, John Wiley & Sons, Inc., New York: 1970.

ARCHIBALD, R. D., *Managing High-Technology Programs and Projects*, New York, Wiley, 1976.

ARDITI, D., ALBULAK, M. Z. *Comparison of Network Analisys with Line of Balance in a Linear Repetitive Construction Project.* Proceedings of the Sixth INTERNET Congress, v. 2, Garmisch-Partenkirchen, W. Germany, Sept., p. 13-25: 1979.

ARDITI, D., ALBULAK, M. Z. *Line of Balance Scheduling in Pavement Construction,* Jornal of Construction Engineering and Management, ASCE, v. 112, n. 3, p. 411-424: 1986.

ARORA, R. K., SACHDEVA. *Distributed Simulation of Resource Constrained Project Scheduling.* Computers & Operations Research, 16, pág. 295-304: 1989.

ASHLEY, D. B. *Simulation of Repetitive Unit Construction.* Journal of Construction Division, ASCE, v. 106, n. CO2, p. 185-194: 1980.

ASHOUR, S. *An Experimental Investigation and Comparative Evaluation of Flow-shop Sequencing Techniques*, Operations Research, v. 18, p. 541-549:1972.

BANDELLONI, M., TUCCI, M., RINALDI, R. *Optimal Resource Leveling Using Non-Serial Dynamic Programming.* European Journal of Operational Research, 78, pág. 162-177: 1994.

BEASLEY, D., BULL, D. R., MARTIN, R. R. *An Overview of Genetic Algorithms: Part I. Fundamentals.* University Computing, n. 15, p. 58-69: 1993.

BEASLEY, D., BULL, D. R., MARTIN, R. R. *An Overview of Genetic Algorithms: Part II. Research Topics.* University Computing, n. 15, p. 170-181: 1993.

BEASLEY, J. E., CHU, P. C. *A Genetic Algorithm for the Set Covering Problem.* European Journal of Operational Research, 94, pág. 392-404: 1996.

BEDAU, M. A., PACKARD, N. H., *Measurement of Evolutionary Activity, Teleology, and Life.* In C. G. Langton, C. Taylor, J.D Farmer S. Rasmussen, eds., Artificial Life II. Adson Wesley :1992.

BEDWORTH, D. D., BAILEY, J. E. *Integrated Production Control Systems - Management, Analisys, Design.* Wiley, New York: 1982.

BELCHIOR, P. G. O. B., *PERT/CPM - Técnica de Avaliação, Revisão e Controle de Projetos.* Edições de Ouro: 1970;

BELL, C. E., PARK, K. *Solving Resource Constrained Project Scheduling Problems by A* Search.* Naval Res. Logist. No. 37. P 61-84: 1990.

BELL, C. E., HAN, J. *A New Heuristic Solution Method in Resouce-Constrained Project Scheduling.* Naval Res. Logist. No. 38. P 315-331: 1991.

BELLMAN, R. ESOGBUE, A. O. e NABESHIMA, I. *Mathematical Aspects of Scheduling and Application.* Pergamon Press: 1982.

BEY, R. P., DOERSCH, R. H., PATTERSON, J. H. *The Net Present Value Criterion: its Impact on Project Scheduling.* Project Management Quart. N. 12, V. 2. P. 35-45: 1981.

BIEGAL, J. E., DAVERN, J. J. *Genetic Algorithm and Job Shop Scheduling.* Computers and Industrial Engineering, v. 19: 1990.

BIRREL, G. S., *Construction Planning - Beyond the Critical Path,* Journal of the Construction Division, ASCE, v. 106, n. CO3, p. 389-407: 1980.

BLAZEWICZ, J., LENSTRA, J. K., RINNOOY KAN, A. H. G. *Scheduling Subject to Resource Constraints: Classification and Complexity,*Discrete Applied Mathematics, 5, pág. 11-24: 1983.

BOITEUX, C. D., *Administração de Projetos: PERT/CPM/ROY*, 5ª. ed., Interciência, Rio de Janeiro: 1979.

BJORNDAL, M. H., CAPRARA, A., COWLING, P. I., DELLA CROCE, F., LOURENÇO, H., MALUCELLI, F., ORMAN, A. J ., PISINGER, D., REGO, C., SALAZAR, J. J. *Some Thoughts on Combinatorial Optimisation.* European Journal of Operational Research. 83. P. 253-270: 1995.

BOCTOR, F. F. *Some Efficient Multi-Heuristics Procedures for Resource-Constrained Project Scheduling.* Eur. J. Opl. Res. No 49. P. 3-13: 1990.

BOCTOR, F. F. *Heuristics for Scheduling Projects with Resources Restrictions and Several Resource-Duration Modes.* International Journal of Production Research, v. 31, n. 11, p. 2547-2558: 1993.

BOCTOR, F. F., *A New and Efficient Heuristic for Scheduling Projects with Resource Restrictions and Multiple Execution Modes.* European Journal of Operational Research. 90. p. 349-361: 1996.

BRANDT, J. D., MEYER, W. L. E SHAFFER, L. R. *The Resouce Scheduling Problem in Construction.* Civil Engineering Studies Report no. 5, Department of Civil Engineering, University of Illinois: 1964.

BURGESS, A. R., KILLEBREW, J. B., *Variations in Activity Level on a Cyclical Arrow Diagram.* Journal of Industrial Engineering, 13(2), pág. 76-83: 1962.

CARR, R. I., MEYER, W. L., *Planning Construction of Repetitive Building Units,* Journal of Construction Division, ASCE, v. 100, n. CO3, p. 403-412: 1974.

CARUANA, R. A., SCHAFFER, J. D. *Representation and Hidden Bias: Gray vs. Bynary Coding for Genetic Algorithms.* In Proceedings of the Fifth International Conference on Machine Learning. Morgan Kaufmann: 1988.

CASAROTO, R. M., *Análise das Curvas de Agregação de Recursos de Pequenos Edifícios em Florianópolis.* Dissertação de Mestrado, EPS/UFSC, Florianópolis: 1995.

CERNY, V. *Thermodynamical Approach to the Traveling Salesman Problem: an Efficient Simulation Algorithm.* Journal of Optimization Theory and Applications. N. 45/1. P. 41-45: 1985.

CHAMBERS, L. *Practical handbook of genetic algorithms: Applications.* CRC Press, Boca Raton: 1995.

CHAN, K. C., TANSRI, H. *A Study of Genetic Crossover Operations on the Facilities Layout Problem*. Computers Industrial Engineering. V. 26. N. 3. P. 537-550: 1994.

CHAN, W., CHUA, D. K. H., KANNAN, G., *Construction Resource Scheduling with Genetic Algorithms*. Journal of Construction Engineering and Management. V. 122(2). P. 125-132: 1996.

CHANG, T. C., IBBS, C. W., *Priority Ranking: a Fuzzy Expert System for Priority Decision Making in Building Construction Resource Scheduling*, Journal of Building and Environment, v. 25, n. 3, p. 253-267: 1990.

CHEH, K. M., GOLDBERG, J. B., ASKIN, R. *A Note on the Effect of Neighborhood Structure in Simulated Annealing*. Computers Operations Research, 18 (6), pág. 537-547: 1991.

CHRAZANWSKY, E. N., JOHNSTON, D., *Application of Linear Scheduling, Journal of Construction Engineering*, ASCE, 112(4), p.476-491: 1986.

CHO, J. H., KIM, Y. D. *A simulated Annealing Algorithm for Resource Constrained Project Scheduling Problems*. Journal of the Operational Research Society. N. 48. P. 736-744: 1997.

CHRISTIAN, J., KALLOURS, G., *Predictive Cost-Time Models for Construction Activities*, CIB, v. 4, p. 157-168: 1990.

CHRISTOFIDES, N., ALVAREZ-VALDES, R., TAMARIT, J. M. *Project Scheduling with Resource Constraints: a Branch and Bound Approach*. European Journal of Operational Research. N° 29. P. 262-273: 1987.

CLEVELAND, G. A., SMITH, S. F. *Using Genetic Algorithms to Schedule Flow Shop Release*, in: Proceedings of the Third International Conference on Genetic Algorithms Applications, pág. 160-169: 1989.

CLOUGH, R. H., SEARS, G. A. *Construction Project Management*. 3rd. Ed., John Wiley & Sons, Inc., New York: 1991.

COLE, L. J. R., *Applied Flow Line Technology*. The Building Economist, n. 15(4), p. 218-224: 1977.

COOPER, D. F. *Heuristics for Scheduling Resource-Constrained Projects: An Experimental Investigation*. Management Science, 22, pág. 1186-1194: 1976.

COOPER, D. F. A Note on Serial and Parallel Heuristics for Resource-Constrained Projects Scheduling. Foundations of Control Engineering, 2, pág. 131-134: 1977.

DANNENBRING, D. G. *An Evaluation of Flow Shop Sequencing Heuristics. Management Science*. N. 23. P. 1273-1283: 1977.

DAVIES, E. M., *An Experimental Investigation of Resource Allocation in Multiactivity Projects*, Operations Research Quarterly, 24 (4), pág. 587-591:1974.

DAVIES, E. M., *New Criterion for Resource Scheduling*, Transportation Engineering Journal, 99, TE4, pág. 741-755: 1973.

DAVIS, L. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, Princeton, N. J.: 1991.

DAVIS, L., *Job Shop Scheduling with Genetic Algorithms. Proceedings of an International Conference on Genetic Algorithms and their Applications*. Ed. Por Grefenstette, J. J., Carnegie-Mellon University, Pittsburgh, Pensylvania, p. 136-140: 1985.

DAVIS, W. E., HEIDRON, G.E. *An Algorithm for Optimal Project Schedulin Under Multiple Resource Constraints*. Management Science, 17 (12), B803-B816: 1971.

DAVIS, W. E., PATTERSON, J. H. *A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling*. Management Science. V. 21. No. 8, p. 944-955: 1975.

DAVIS, W. E., *Project Scheduling Under Resources Constraints-Historical Review and Categorization of Procedures*. AIIE Transactions. V. 5(4), p. 147-163: 1973.

DAVIS, W. E., *Project Summary Measures and Constrained Resouce Scheduling*, AIIE Transactions, 7 (2), pág. 132-142: 1975.

DAWOOD, N. N., *Developing a Production Management Modelling Approach for Precast Concrete Building Products*, Construction Management and Economics, v. 12, UK, p. 393-412: 1994.

DELLA CROCE, F., TADEI, R., VOLTA. *A Genetic Algorithm for the Job Shop Problem*. Computers Operations Research. v. 22. n. 1. p 15-24: 1995.

DEMEULEMEESTER, E. L., DODIN, B., HERROELEN, W., A Random Activity Network Generator. Operations Research, V. 41, No. 5, pág. 972-980: 1993.

DEMEULEMEESTER, E. L., HERROELEN, W. S. *An Efficient Optimal Solution Procedure for the Preemptive Resource-Constrained Scheduling Problem*. European Journal of Operational Research. 90, pág. 334-348: 1996.

DE WERRA, D., HERTZ, A. *Tabu Search Techniques: a Tutorial and an Application to Neural Networks*. Operations Research Spektrum. N. 11. P. 131-141: 1989.

DOERSCH, E. H., PATTERSON, J. H. *Scheduling a Project to Maximize its Present Value: a Zero-One Programming Approach*. Management Science, 23 (8), 882-889: 1977.

DRESSLER, J., *Stochastic Scheduling of Linear Construction Sites*. Journal of Construction Division, ASCE, v. 100, n. CO4, p. 571-588: 1974.

DREXEL, A., GRUENEWALD, J. Nonpreemptive Multi-Mode Resource-Constrained Project Scheduling. IIE Transactions, 25(5), pág. 74-81: 1993.

EASA, S. M., Resource Leveling in Construction by Optimization. Journal of Construction Engineering and Management, 115(2), pag. 302-316: 1989.

EGLESE, R. W. *Simulated Annealing: A Tool for Operational Research.* European Journal of Operational Research. 46. P. 271-281: 1990.

EGLESE, R. W., RAND, G. K. *Conference Seminar Timetabling.* European Journal of Operational Research. 38. P. 591-598: 1987.

ELDIN, N. N., SENOUCI, A. B., *Scheduling of Control of Linear Projects*. Canadian Journal of Civil Engineering, 21(2), p. 219-230: 1994.

ELMAGHRABY, S. E., *Activity Networks: Project Planning and Control by Network Models*, Wiley, New York: 1977.

ELMAGHRABY, S. E., HERROELEN, W. S., *On the Measurement of Complexity in Activity Networks*. European Journal of Operations Research, 5, pág. 223-234:1980.

ELMAGHRABY, S. E. *Project Bidding Under Deterministic and Probabilistic Activity Durations*. European Journal of Operational Research. N. 49. P. 14-34: 1990.

ELMAGHRABY, S. E., HERROELEN, W. S. *The Scheduling of Activities to Maximize the Net Present Value of Projects*. European Journal of Operational Research. N. 49. P. 35-49: 1990.

ELSAYED, E. A. *Algorithms for Project Scheduling with Resource Constraints.* International Journal of Production Research, 20, pág. 95-103: 1982.

EPSTEIN, S., WILAMOWISKY, Y., DICKMAN, B. *Deterministic Multiprocessor Scheduling With Multiple Objectives*. Computers Operations Research, 19 (8), pág. 743-749: 1992.

ESHELMAN, L. J., SCHAFFER, J. D. *Preventing Premature Convergence in genetic Algorithms by Preventing Incest*. In R. K. Belew and L. B. Booker, eds., Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann: 1991.

EVANS, J. R., *Structural Analysis of Local Search Heuristics in Combinatorial Optimization*. Computers & Operations Research, n. 14, p. 465-477: 1987.

FENG, C-W, LIU, L., BURNS, S. A., *Using Genetic Algorithms To Sove Construction Time-Cost Trade-Off Problems.* Journal of Computing Engineering, V. 11, 3, 184-190: 1997.

FERREIRA, A. B. H., *Novo Dicionário da Língua Portuguesa.* 2ª ed, Nova Fronteira: Rio de Janeiro: 1986.

FORBES, W. S., *Flow Charts to Control Progress on Housing Sites.* Building Research Station Digest, n. 134, oct.: 1971.

FOURMAN., M. P . *Compaction of Symbolic Layout Using Genetic Algorithms.* Proc. Int. Conf. Genetic Algorithms Appl., p. 141-153: 1985.

FULKERSON, D. R., *A Network Flow Computation for Project Cost Curves,* Management Science, v. 7, n. 2: 1961.

GATES, M., SCARPA, A., *Conceptual RMC/Time Synthesis,* Journal of Construction Division, n. 6, p. 307-323: 1976.

GAUTHIER, F. A. O. *Programação da Produção: uma Abordagem Utilizando Algorítmos Genéticos.* Tese de Doutorado. UFSC. Florianópolis: 1993.

GLOVER, F. *Future Paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research.* N. 13. P. 533-549: 1986.

GLOVER, F. *Tabu Search: Part I.* ORSA Journal on Computing. N.1. p. 190-206: 1989.

GLOVER, F. *Tabu Search: Part II.* ORSA Journal on Computing. N.2. p. 4-12: 1990.

GLOVER, F., GREENBERG, H. J. *New Approaches dor Heuristic Search: a Linkage with Artificial Intelligence.* European Journal of Operational Research, 39 (5), p. 119-130: 1989.

GLOVER, F., McMILLAN, C., NOVICK, B. *Interactive Decision Software and Computer Graphics for Architetural and Space Planning.* Annals of Operations Research. N.5. p. 557-573: 1985.

GOLDBERG, D. E. *A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-oriented Simulated Annealing.* Complex Systems, 4, pág. 445-460: 1990.

GOLDBERG, D. E. *Computed-Aided Pipeline Operation Using Genetic Algorithms and Rule Learning.* API Pipeline Cybernetics Symp., Houston, Texas, TX: 1984.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, New York: 1989.

GOLDBERG, D. E., LINGLE, Jr. R. *Alleles, Loci and the Travelling Salesman Problem*. Proceedings of an Carnegie-Melon University, Pittsburgh, Pennsylvania, p. 154-159: 1985.

GREFENSTETTE, J. J., *Optimization of Control Parameters for Genetic Algorithms*. IEEE Transactions on Systems, Man, and Cybernetics, 16, n. 1, p. 122-128: 1986.

GREFENSTETTE, J. J., GOPAL, R., ROSMAITA, B., VAN GUCHT, D., *Genetic Algorithm for the Travelling Salesman Problem*. Proceedings of an Carnegie-Melon University, Pittsburgh, Pennsylvania, p. 160-169: 1985.

GRINOLD, R. G. *The Payment Schedule Problem*. Naval Research Logist. Q. No 19. P. 123-136: 1972.

HANDA, V. K., BARCIA, R. M., *Linear Scheduling Using Optimal Control Theory*, J. Const. Engrg., ASCE, v. 112, n. 13, p. 387-393: 1986.

HARBERT, J. A., *Development of the Combined PERT and LOB (CPL) Chart*. Proceedings of the Fifth Internet Congress, v. Friday, Birmingham, U. K., p. 241-246: 1976.

HARDING, H. A., *Administração da Produção*, Atlas, São Paulo: 1981.

HARRIS, F., McCAFFER, R., *Modern Construction Management*, Crosby Lockwood Staples, London, U.K.: 1977.

HARRIS, R. B. *Packing Method for Resource Leveling (Pack)*, Journal of Construction Engineering and Management, 116(2), pág 331-350: 1990.

HARVEY, R. T., PATTERSON, J. H. *An Implicit Enumeration Algorithm for the Time/ Cost Tradeoff Problem in Project Network Analysis*, Foundations of Control Engineering, 4, pág. 107-117: 1979.

HASTINGS, N. A J. *On Resource Allocation in Project Networks*. Journal of the Operational Research Society, 23, pág. 217-221: 1972.

HEINECK, L. F. M. *Curvas de Agregação de Recursos no Planejamento e Controle da Edificação - Aplicações a Obras e a Programas de Construção.* Caderno de Engenharia CE-31/89. Curso de Pós-Graduação em Engenharia Civil / UFRS. Porto Alegre: 1989.

HEINECK, L. F. M., *Inventário de Aplicações da Curva S no Gerenciamento de Produção Civil: uma Aplicação no Controle de Empreendimentos*, UFMG, 10° ENEGEP, v. 2, p. 736-741: 1990.

HERROELEN, W. S., GALLENS, E. *Computational Experience with an Optimal Procedure for the Scheduling of Activities to Maximize the Net Present Value*. European Journal of Operational Research. N. 65. P. 274-277: 1993.

HERTZ, A. *Tabu Search for Large Scale Timetabling Problems*. European Journal of Operational Research. N. 54. P. 39-47: 1994.

HERTZ, A., De WERRA, D. *The Tabu Search Metaheuristic: how we used it*. Annals of Mathematics and Artificial Intelligence. N. 1. P. 111-121: 1990.

HINDERLANG, T. J. MUTH, J. F. *A Dynamic Programming Algorithm for Decision CPM Networks*, Operations Research, 27, pág. 225-241: 1979.

HOLLAND, J. H . *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor: 1992.

ICMELI, O., ERENGUC, S. S. *A Tabu Search Procedure for the Resource Constrained Project Scheduling Problem with Discounted Cash Flows*. Computers and Operations Research. N. 21. P. 841-853: 1994.

ICMELI, O., ERENGUC, S. S. *A Branch and Bound Procedure for the Resource Constrained Project Scheduling Problem with Discounted Cash Flows*. Management Science, v. 42, n. 10, p. 1395-1408: 1996.

ISHIBUSHI, H., MISAKI, S., TANAKA, H. *Modified Simulated Annealing for the Flow Shop Sequencing Problem*. European Journal of Operational Research. N. 81. P. 388-398: 1995.

JANIKOW, C. Z., MICHALEWICZ, Z. *An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms*. In R. K. Belew and L. B. Booker, eds., Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann: 1991.

JEFFCOAT, D. E., BULFIN, R. L. *Simulated Annealing for Resource-Constrained Scheduling*. European Journal of Operational Research. N. 70. P. 43-51: 1993.

JOHNSON, D. A., ARAGON, C. R., McGEOCH, L. A., SCHEVON, C. *Optimization by Simmulated Annealing: An Experimental Evaluation. Part I, Graph Partioning*. Operations Research. V. 37. No. 6. p. 865-892: 1989.

JOHNSTON, D. W., *Linear Schedule Method for Highway Construction*, Journal of Construction Division, ASCE, v. 107, n. CO2, p. 247-261: 1981.

KAIMANN, R. A., *Coefficient of Network Complexity*, Management Science, 21 (2), pág. 172-177: 1974.

KAKA, A. P., PRICE, A. D. F., *Modelling Standard Cost Commitment Curves for Contractors Cash Flow Forecasting*, Construction Management and Economics, n. 11, p. 411-420: 1993.

KAPSALIS, A., RAYWARD-SMITH V. J., SMITH G. D. *Solving the Graphical Steiner Tree Problem Using Genetic Algorithms*. Journal of the Operational Research Society, 44 (4), pág. 397-406: 1993.

KAZAZ, B., SEPIL, C. *Project Scheduling with Discounted Cash Flows and Progress Payments*. Journal of the Operational Research Society. N. 47. P. 1262-1272: 1996.

KELLEY, J. E., *Critical Path Planning and Scheduling Mathematical Bases*, Operations Research, v. 9, n. 3, 1961.

KELLEY, J. E., *The Critical Method: Resources Planning and Scheduling. In* Muth and G.L. Thompsom, Industrial Scheduling, Prentice-Hall, pág. 347-365. New Jersey: 1963.

KHATTAB, M., CHOOBINEH, F. *A New Heuristic for Project Scheduling With a Single Resource Constraint*. Comp. Ind. Eng., 19, pág. 514-518: 1990.

KHATTAB, M. M., CHOOBINEH, F. *A New Approach for Project Scheduling with a Limited Resource*. Int. J. Prod. Res. No 30: 185-198: 1991.

KHISTY, C. J. *The Application of the Line of Balance Technique to the Construction Industry*. Indian Concrete Journal, v. 100, n. 3, p. 403-412: 1970.

KIRKPATRICK, S., GELLAT Jr., C. D., VECCHI, M. P. *Optimization by Simulated Annealing*. Science. N. 220/4598. P. 671-680: 1983.

KLEINFELD, I. H . *Manpower use in High-Hise Residential Construction*, Journal of Construction Division, ASCE, v. 102, n. CO2, p. 379-383: 1976.

KOLISCH, R., *Project Scheduling Under Resource Constraints - Efficient Heuristics for Several Problem Classes*, Physica, Heidelberg: 1995.

KOLISCH, R., *Serial and Parallel Resource-Constrained Project Scheduling Methods Revisited: Theory and Computation*. European Journal of Operational Research. N. 90, p. 320-333: 1996.

KRONE, M. J., STEIGLITZ, K. *Heuristic Programming Solution of a Flow Shop Scheduling Problem*. Operations Research. N. 22. P. 629-638: 1974.

KURTULUS, I. S., DAVIS, E. W. *Multi-Project Scheduling: Categorization of Heuristic Rules Performance*. Management Science, 28, pág. 161-172: 1982.

KURTULUS, I. S., NARULA, S. C. Multi Project Scheduling: Analysis of Project Performance. IIE Transactions, 17, pág. 58-66: 1985.

LEVITIN, G., RUBINOVITZ, J. *Genetic Algorithm for Linear and Cyclic Assignment problem*. Computer Ops Res., v. 20, n. 6, p. 597-586: 1993.

LEVINE, H. A., ALIBERTI, E. M., FORD, B. P., *The Application of Line of Balance on an International Project*, Proceedings of the Fifth Internet Congress, v. Thursday, Birmingham, U.K., p. 251-258: 1976.

LEVY, F. K., THOMPSOM, G. L., WIEST, J. D. Multiship, Multishop, Workload-Smothing Program, Naval Research Logistics Quarterly, 9, pág. 37-44: 1962.

LI, K. Y., WILLIS, R. J. *An Iterative Scheduling Technique for Resource-Constrained Project Scheduling.* European Journal of Operational Research. N. 56. P. 370-379: 1992.

LI, S., *New Approach for Optimization of Overall Construction Schedule*. V. 122, 1, pág. 7-13: 1996.

LIEPINS, G. E., HILLIARD, M. R. *Genetic Algorithms: Foundations and Applications.* Annal Opns. Res., n. 21, p. 31-58: 1989.

LIMMER, C. V., *Planejamento, Orçamentação e Controle de Projetos.* Caderno de Produção Civil 16/91, Universidade Federal Fluminense, Pós-Graduação em Engenharia Civil, Niteroi: 1991.

LEE, J. K., KIM, Y. D. *Search Heuristics for Resouce-Constrained Project Scheduling.* Journal Opl. Res. Soc. N. 47. P. 678-689: 1996.

LEON, V. J., BALAKRISHNAN, R. *Strength and Adaptability of Problem-Space Based Neighborhoods for Resource Constrained Scheduling*, OR Spektrum, 17, pág. 173-182: 1995.

LOPEZ VACA, O. C. *Um Algoritmo Evolutivo para a Programação de Projetos Multi-Modos com Nivelamento de Recursos Limitados.* Tese de Doutorado. UFSC. Florianópolis: 1995.

LUMSDEN, P., *The Line of Balance Method*, Pergamon Press, London, U.K.:1968.

LUNDY, M., MEES, A. *Convergence of an Annealing Algorithm*. Mathematical Programming. N. 34. P. 111-124: 1986.

LUTZ, J. D., HIJAZI A. *Planning Repetitive Construction: Current Practice.* Construction Management and Economics, 11,pág. 99-110: 1993.

MACCARTHY, B. L. LIU, J. *Addressing the Gap Scheduling Research: a Review of Optimization and Heuristic Method in Production Scheduling.* International Journal of Production Research. v. 31, n. 1, pp. 59-79: 1993.

MANGIN, J. C. *Scheduling Methods for Repetitive Tasks in Second Stage Building Construction*, Proceedings of the Sixth Internet Congress, Book II, Garminsch-Partenkirchen, W. Germany, p. 287-295: 1979.

MARSHALL, C. W., *Appied Graph Theory*. John Wiley and Sons, New York: 1971.

MATTILA, K. G., ABRAHAM, D. M., *Resource Leveling of Linear Schedules Using Linear Programming*. V. 124, 3, pág. 232-244: 1998.

MATSUO, H., SUH, C. J., SULLIVAN, R. S. *A Controlled Search Simulated Annealing Method for the Single-Machine Weighted Tardiness Problem*. Annals of Operations Research. N. 21. P. 85-108: 1989.

MAWDESLEY, M. J., ASKEW, W. H., LEES, J., TAYLOR, J., STEVENS, C., *Time Change Charts for Scheduling Linear Projects*. Computing in Civil Engineering, ASCE, p. 613-620: 1990.

MAYHUGH, J. O. *On the Mathematical Theory of Schedules*. Management Science, 11(2), pág. 289-307: 1964.

MEYER, W. L. and SHAFFER, L.R. *Extensions of the critical path method through the application of integer programming*. Civ. Engrg. Constr. Res. Ser. 2, Univ. of Illinois, Urbana, III: 1963.

MELIN, J. W., *Fenced Bar Charts for Repetitive Projects. Paper* apresentado no ASCE Spring Convention, Atlanta, Ga.: 1984.

MELIN, J. W., WHITEAKER, B., *Fencing a Bar-Chart*, Journal of Construction Division, ASCE, v. 107, n. CO3, p. 497-507: 1981.

MITCHELL, M., *An Introduction to Genetic Algorithms*. MIT, Cambridge: 1996.

MODER, J. J., PHILLIPS, C. R., DAVIS, E. W. Project Management With CPM, PERT and Precedence Diagramming. 3ª ed., Reinhold, New York: 1983.

MOHANTHY, R. P., SIDDIQ, M. K. *Multiple Projects Multiple Resouces-constrained Scheduling: Some Studies*. International Journal of Production Research, 27 (2), pág. 261-280: 1989.

MORAD, A., BELIVEAU, Y., *Geometric Based Reasoning System for Project Planning*. Journal Comp. In Civil Engrg, ASCE, v. 8, n. 1, p. 52-71: 1993.

MOSHELHI, O., *Scheduling of Repetitive Projects with Cost Optimization*. Journal of Construction Engineering and Management, v. 119, n. 4, p. 681-697: 1993.

MÜLLER-MERBACH, H. *Heuristics and their Design: a Survey*. European Journal of Operational Research, n. 8, p. 1-23: 1981.

NEALE, R. H., NEALE, D. E. *Construction Planning*. 1ª ed., Thomas Telford Ltda., London: 1989.

NORBIS, M. I., SMITH, J. M. *A Multi-Objective Multi-Level Heuristic for Dynamic Resource-Constrained Scheduling Problems*. European Journal of Operational Research, n. 33, p.30-41: 1988.

NORBIS, M. I., SMITH, J. M. *Two Level Heuristic for the Resource Constrained Scheduling Problem*. International Journal of Production Research, 24 (5), pág. 1203-1219: 1986.

NOWICKI, E., SMUTNICKI, C. *A Fast Taboo Search Algorithm for the Job Shop Problem*. Management Science. V. 42. N. 6. P. 797-813: 1996.

O'BRIEN, J. J., *Network Scheduling Variations for Repetitive Work*, Paper Apresentado no ASCE Spring Convention, Atlanta, Ga: 1984.

O'BRIEN, J. J., KREITZBERG, F. C., MIKES, W. F., *Network Scheduling Variations for Repetitive Work*, Jounal of Construction Engineering and Management, ASCE, v. 111, n. 2, p. 105-116: 1985.

O'BRIEN, J. J., *VPM Scheduling for High Rise Buildings*, Journal of the Construction Division, ASCE, v. 101, n. CO4, p. 895-905:1975.

O'BRIEN, J. J., *Scheduling Handbook*. McGraw-Hill. New York: 1969.

OGBU, F. A., SMITH, D. K. *The Application of the Simulated Algorithm to the Solution of the n/m/Cmax Flow Shop Problem*. Computers and Operations Research. N. 17. P. 243-253: 1990.

OGUZ, O., BALA, H. *A Comparative Study of Computational Procedures for the Resource Constrained Project Scheduling Problem*. European Journal of Operational Research. N. 72. P. 406-416: 1994.

OLIVER, I. M., SMITH, D. J., HOLLAND, R. C. *A Study of Permutation Crossover Operations on the Travelling Salesman Problem*. Proc. Of the 2nd. Int. Conf. Of Genetic Algorithms. P. 224-230: 1987.

O'REILLY, U. M., OPPACHER, F. *The Troubling Aspects of a Building Block Hypothesis for Genetic Programming*. In L. D. Whitley and M. D. Vose, eds., Foundations of Genetic Algorithms 3. Morgan Kaufmann: 1995.

OSMAN, I. H., POTTS, C. N. *Simulated Annealing for Permutation Flow-Shop Scheduling*. OMEGA International Journal of Management Science. N. 17/6. P. 551-557: 1989.

OXLEY, R., POSKITT, J., *Management Techniques Applied to the Construction Industry*, 2ª. ed, Crosby Lockwood, U.K.: 1971.

ÖZDAMAR, L. ULUSOY, G. *A Local Constraint Based Analysis Approach to Project Scheduling Under General Resource Constraints.* European Journal of Operational Research. N. 79. P. 287-298: 1994.

PADMAN, R. SMITH-DANIELS, D. E. *Early-Tardy Cost Trade-Offs in Resource Constrained Projects with Cash Flows: an Optimization-Guided Heuristics Approach.* European Journal of Operational Research. N. 64. P. 295-311: 1993.

PAPADIMITRIOU, C. H., STEIGLITZ, K. *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NY: 1982.

PARKER, R. G., RARDIN, R. L., *An Overview of Complexity Theory in Discrete Optimization: Part I. Concepts.* IIE Transactions. V. 14. No. 1: 1981.

PARKER, R. G., RARDIN, R. L., *An Overview of Complexity Theory in Discrete Optimization: Part II. Results and Implications.* IIE Transactions. V. 13. No. 1: 1982.

PASCOE, T. L., *Allocation of Resources CPM.* Revue Francaise Recherche Operationalle, 38, 31-38: 1966.

PATTERSON, J.H. *Alternate Methods of Project Scheduling With Limited Resources.* Naval Research Logistics Quarterly, 20, pág. 767-784: 1973.

PATTERSON, J.H. *Project Scheduling: the Effects of Problem Structure on Heuristic Performance.* Naval Research Logistics Quarterly, 23, pág. 95-123: 1976.

PATTERSON, J. H., HUBER, W. D. *A Horizon-Varying, Zero-One Approach to Project Scheduling.* Management Science, 20 (6), 990-998: 1974.

PEDERSEN, H., *Network Planning of Repetitive Processes in Housing Construction Industry*, Proceedings of the Third Internet Congress, Book II, p. 381-392, Estocolmo, Suécia: 1972.

PEER, S., *Application of Cost-Flow Forecasting Models.* Journal of Construction Division, n. 6, v. 108, 1982.

PERERA, S. *Resource Sharing in Linear Construction.* Journal of Construction Engineering and Management, ASCE, 109 (1), pág. 102-111: 1983.

PERERA, S., *Linear Programming Solution to Network Compression*, Journal of the Construction Division, ASCE, v. 106, n. CO3, p. 315-326 :1980.

PERERA, S., *The Chain Bar-Chart for Project Planning and Control*, National Development, june/july, p. 68-72: 1981.

PERERA, S., *Network Planning of Project Comprising Repetitive Activities*, IAHS Conference on the Impact of Economy and Technology, p. 827-985, Viena, Austria: 1982.

PERRY, W. W., *Automation in Estimating Contractor Earnings*, Military Engineer, nov/dez, p. 393-395: 1970.

PHILLIPS, S., DESSOUKI, M. I. *Solving the Project Time/Cost Tradeoff Problem Using the Minimal Cut Concept.* Management Science, 24 (4), 393-400: 1977.

PIERREVAL, H., TAUTOU, L., *Using Evolutionary Algorithms and Simulation for the Optimization of Manufacturing Systems.* IIE Transactions, 29, pág. 181-189: 1997.

PILCHER, R. *Principles of Construction Management.* 2nd. Ed., McGraw-Hill Book Company, London: 1976.

POPESCU, C., *A Planning Method for Linear Projects.* Proceedings of the Management Institute, Seminar Symposium , Atlanta, Georgia, USA: 1979.

PRITSKER, A. B., WATTERS, L. J., e WOLFE, P. M. *Multi-project Scheduling with Limited Resources: a Zero-One Programming Approach.* Management Science, 16 (1), 93-109: 1969.

PRÜGEL-BENNET, A., SHAPIRO, J. L., *An Analysis of Genetic Algorithms Using Statistical Mechanics.* Physical Review Letters 72, no 9, p. 1305-1309: 1994.

RAHBAR, F. F., ROWING, J. E. *Repetitive Activity Scheduling Process.* AACE Trans., 36th Annual Meeting, Orlando: 1992.

REDA, R. M., RPM: *Repetitive Project Modeling. Journal of Construction Engineering and Management*, ASCE, 116(2), p. 316-330: 1990.

REEVES, C. R., *A Genetic Algorithm for Flowshop Sequencing.* Computers Operations Research. v. 22. n. 1. p 5-13: 1995.

REEVES, C. R., *An Introduction to Genetic Algorithms. In OR Tutorial Papers.*(A. G. Munford e T. C. Bailey, eds.). Operational Research Society, p. 69-83: 1991.

REEVES, C. R., *Modern Techniques for Combinatorial Problems.* John Wiley & Sons, Inc. New York, N.Y.:1993.

RIST P. T. *Cascade Charts in Construction.* Proceedings of the Third INTERNET Congress, Book II, pág. 415-422, Stockholm, Sweden :1972.

ROBINSON, D. R., *A Dynamic Programming Solution to Cost-Time Trade Off for CPM*, Management Science, v. 22, n. 2: 1975.

ROESCH, W., *Network Planning and Velocity-Diagrams in Housing Construction Industry*. Proceedings of the Third Internet Congress, Book II, p. 415-422, Stocolmo, Suécia: 1972.

RUSSEL, A. H. *Cash Flows in Networks*. Management Science. N. 16. P. 357-373: 1970.

RUSSEL, A. H. *A Comparison of Heuristics for Scheduling Projects with Cash Flows and Resource Restrictions*. Management Science. N. 32. P. 1291-1300: 1986.

RUSSEL, A. D., CASELTON, W. F., *Extensions to Linear Scheduling Optimization*. Journal of Construction Engineering and Management, ASCE, 114(1), p. 36-52: 1988.

RUSSEL, D., *Extensions to Linear Scheduling Optimization*, Journal of Construction Engineering and Management. V. 114, n. 1, p. 36-52: 1988.

RUSSEL, A. D., WANG, W. C. M. *New Generation of Planning Structures*. J. Const. Engrg. And Mgmt., ASCE, 119(2), pág. 196-214: 1993.

SAMPSON, S. E., WEISS, E. N. *Local Search Techniques for the Generalized Resource Constrained Project Scheduling Problem*. Naval Research Logistics. N. 40. P. 665-675: 1993.

SCHAFFER, J. D., CARUANA, R. A., ESHELMAN, L. J., DAS, R., *A Study of Control Parameters Affecting Online Performance of Genetic Algorithm for Function Optimization*. Proceedings of the Third International Conference in Genetic Algorithms:1989.

SCHAFFER, J. D., GREFENSTETTE, J. J. *Multi-Objetive Learning via Genetic Algorithm*. Proc. 9th Int. Conj. Artif. Intel., Los Angeles, p. 593-595: 1985.

SCHODERBEK, P. P., DIGMAN, L. A., *Third Generation PERT/LOB*, Harvard Business Review, v. 45, n. 5, p. 100: 1967.

SCHRAGE, L. *Solving Resource-Constrained Network Proms by Implicit Enumeration-non-preemptive Case*, Operations Research, 18 (2), pág. 225-235: 1970.

SCHRAGE, L. *Solving Resource-Constrained Network Problems by Implicit Enumeration, Preemptive Case*. Operations Research, 20 (3), 668-677: 1972.

SELINGER, S., *Construction Planning for Linear Projects*, Journal of the Construction Division, ASCE, v. 106, n. CO2, p. 195-205: 1980.

SENOUCI, A. B., ELDIN, N. N., *Dynamic Programming Approach to Scheduling of Nonserial Linear Projects*. Journal of Computing in Civil Engineering, v. 10, n. 2, p. 106-114: 1996.

SLOWINSKI, R. *Two Approaches to Problems of Resource Allocation Among Project Activities: A Comparative Study.* Journal of the Operational Research Society, 31 (8), pp. 711-723: 1980.

SLOWINSKI, R. *Multiobjective Network Scheduling with Efficient Use of Renewable and Non-Renewable Resources.* European Journal of Operational Research, 7 (3), 265-273: 1981.

SHAFFER, L. R., RITTER, J. B., MEYER, W. L. *The Critical Path Method.* McGraw-Hill, New York: 1965.

SHAKED, O., WARSZAWSKI, A., *Knowledge-Based System for Construction Planning of High-Rise Buildings*, Journal of Construction Engineering and Management, v. 121, n. 2, p. 172-182: 1995.

SHTUB, A., BARD, J. F., GLOBERSON, S., *Project Management: Engineering, Technology and Implementation*, New Jersey, Prentice Hall: 1994.

SHTUB, A. LeBLANC, L. CAI, Ziyong. *Scheduling Programs With Repetitive Projects: A Comparison of a Simulated Annealing, a Genetic and a Pair-Wise Swap Algorithm*. European Journal of Operational Research, 88, pp. 124-138: 1996.

SILVER, E. A., VIDAL, R. V. V., De WERRA, D.; *A Tutorial on Heuristic Methods*, European Journal Operational Research, n. 5, p. 153-162: 1980.

SMITH, S. F. *Flexible Learning of Problem Solving Heuristics through Adaptative Search*. Proc. 8th Int. Joint Conf. Artif. Intel., p. 422-425: 1983.

SMITH-DANIELS, D. E., AQUILANO, N. J. *Using a Late Start Resource Constrained Project to Improve Project Net Present Value*. Decision Science. N. 17. P. 617-630: 1987.

SMITH-DANIELS, D. E., SMITH-DANIELS, N. J. *Maximizing the Present Value of a Project Subject to Materials and Capital Constraints.* Journal Opns. Mgmt.. N. 7. P. 33-46: 1987.

STORER, H. R., WU, S. W., VACCARI, R. *New Search Spaces for Sequencing Problems With Application to Job Shop Scheduling.* Management Science, v. 38. n. 10, p. 1495-1509: 1992.

STRADAL, O., CACHA, J., *Time Space Scheduling Method*, Journal of Construction Engineering, ASCE, 108(3), p. 445-457: 1982.

STREIM, H., *Heuristische Lösungsverfahren - Versuch einer Begriffsklärung*, Z. Operations Res., n. 19, p. 143-162: 1975.

SUHAIL, S. A., NEALE, R. H., CPM/LOB*: New Methodology to Integrate CPM and Line of Balance*. Journal of Construction Engineering and Management, ASCE, 120(3), p. 667-684: 1994.

SUHAIL, S. AA. *Out-of-Logic Progress*. Cost Engineering, American Association os Cost Engineers, 35(4), pág. 23-28: 1993.

TALBOT, F. B. *Resource-Constrained Project with Time-Resouce Trade-Offs: The Non-Preemptive Case*. Management Science, 28 (10), 1197-1210: 1982.

TALBOT, F. B., PATTERSON, J. H. *An Efficient Integer Programming Algorithm with Network Cuts for Solving Resource-Constrained Sequencing Problems*. Management Science, 24 (11), p. 1163-1174: 1978.

TAVARES, L. V. *Optimal Resource Profiles for Program Scheduling*. European Journal of Operational Research, 29, pág. 83-90: 1987.

THABET, W., BELIVEAU, Y., *HVLS: Horizontal and Vertical Logic Scheduling for Multistory Projects*, Journal of Construction Engineering and Management, ASCE, v. 120, n. 4, p. 895-892: 1994.

THABET, W., BELIVEAU, Y., *Modeling Work Space to Schedule Repetitive Floors in Multi-Story Buildings*, Journal of Construction Engineering and Management, ASCE, v. 120, n. 1, p. 96-116: 1993.

THABET, W. Y., BELIVEAU, Y. J., *SCaRC: Space-Constrained Resource-Constrained Scheduling System*, Journal of Computing in Civil Engineering, v. 11, n. 1, p. 48-59: 1997.

THESEN, A. *Heuristic Scheduling of Activities Under Resource and Precedence Restrictions*. Management Science, 23, pág. 412-422: 1976.

THESEN, AA., *Measures of the Restrictiveness of Projects Networks*, 7 (3), pág. 193-208: 1977.

TRIMBLE, G. *Resource-Oriented Scheduling*. Project Management Journal, Project Management Institute, 2(2), 70-74: 1984.

ULUSOY, G., OZDAMAR, L. *Heuristic Performance and Network/Resource Characteristics in Resource-constrained Project Scheduling*, Journal of the Operational Research Society, 40, pág. 1145-1152: 1989.

VAN LAAHOVEN, P. J. M., AARTS, E. H. L. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, Dordrecht: 1987.

VENUGOPAL, V., NAREDRAN, T. T. *A Genetic Algorithm Approach to the Machine-component Grouping Problem With Multiple Objectives*. Computers and Industrial Engineering, 22 (4), pág. 469-480: 1992.

VORSTER, M. C., BAFNA, T., *Discussion of Formal Development of Line-of Balance Technique*, Journal of Construction Engineering and Management, ASCE, v. 118, n. 1, p. 210-211: 1992.

WALKER, G. S., *Learning Curve Theory Applied to Networking a Construction Project*, Proceedings of the Sixth Internet Congress, Book II, p. 515-520: 1979.

WEGLARZ, J., BLAZEWICZ, J., CELLARY, J., SLOWINSKI, R. *An Automatic Revised Simplex Method for Constrained Network Scheduling*. ACM Transactions on Mathematical Software, 3, pág. 295-300: 1977.

WEGLARZ, J. *Control in Resource Allocation Systems*. Foundation of Control Engineering, 5 (3), 159-180: 1980.

WEGLARZ, J. *Project Scheduling with Continously Divisible Double Constrained Resources*. Management Science, 27 (9), 1040-1053: 1981.

WHITEHOUSE, G. E., BROWN, J. R. Genres: *An Extension of Brooks Algorithm for Project Scheduling with Resource Constraints*. Computers & Industrial Engineering, 3, pág. 261-268: 1979.

WHITLEY, D., STARKWETHER, T., SHANER, D., *The Travelling Salesman and Sequence Scheduling Problems: Quality Solutions Using Genetic Edge Recombinations*, Handbook of Genetic Algorithms, Editado por Davis, L., Van Nostrand Reinhold, New York, p. 350-372: 1991.

WIEST, J. D. *A Heuristic Model for Scheduling Large Projects with Limited Resources*. Management Science, 13 (6), B359-B377: 1967.

WILLIS, R. J. Critical Path Analysis and Resource Constrained Project Scheduling - Theory and Practice. Opl. Research, 21, pág. 149-155:1985.

WRIGHT, M. B. *Applying Stochastic Algorithms to a Locomotive Scheduling Problems*. Journal of the Operational Research Society. N. 40. P. 101-106: 1988.

YANNAKAKIS, M. *The Analisys of Local Search Problems and their heuristics*, in: Lecture Notes in Computer Science, v. 415, Springer-Verlag, Berlin, p. 298-391: 1990.

YANG, T., IGNIZIO, J. P. *An Exchange Heuristic Algorithm for Project Scheduling with Limited Resources*. Enging. Opt. N. 14. P. 189-205: 1989.

YANG, K. K., TALBOT, F. B., PATTERSON, J. *Scheduling a Project to Maximize its Present Value: an Integer Programming Approach*. European Journal of Operational Research. N. 64. P. 188-198: 1993.

ZOUEIN, P. P., TOMMELEIN, I. D., *Space Schedule Construction*, Proc. 5th Int. Conf. In Civil and Build. Engrg., ASCE, New York, N. Y.: 1993.

ZOZAYA-GOROSTIZA, C., HENDRICKSON, C., REHAC, D. R., *A Knowlwdge-Intensive Planner for Construction Projects*, Journal Build. and Environment, v. 25, n. 3, p. 269-278: 1990.

ANEXO

Nesta seção encontram-se os problemas utilizados para testar o protótipo computacional do método de solução proposto. Estes problemas foram extraídos de trabalhos acadêmicos de pós-graduação, realizados sobre construções reais. Alguns dados, como o número máximo de oficiais previsto para cada atividade ou as despesas diretas associadas às atividades, não estavam disponíveis em alguns trabalhos, e tiveram que receber valores não originais; no entanto, este fato não compromete a qualidade dos experimentos.

DADOS DOS PROBLEMAS

No grupo de problemas, os dados iniciais são relativos ao tipo de rede (mista ou linear) e ao número de pavimentos repetitivos.

Na tabela *1* estão os dados relativos aos períodos de tempo e às disponibilidades de recurso monetário. Cada período (mês) é identificado numericamente na primeira coluna, e cada quantidade de dias pertinentes encontra-se na segunda coluna. Na terceira coluna encontra-se o recurso monetário disponível para cada período em evidência.

Os dados das atividades estão presentes nas tabelas *2-a* e *2-b*. Nas duas primeiras colunas de *2-a* pode-se ver a identificação numérica e denominação, respectivamente; na terceira coluna estão as precedências diretas. As duas primeiras colunas de *2-b* são de identificação numérica e de tipo (**R**epetitiva ou **N**ão **R**epetitiva), respectivamente; a terceira coluna contém o sentido de execução (**A**scendente ou **D**escendente); a quarta e a quinta coluna referem-se aos números mínimo e máximo de equipes; a sexta coluna contém a duração máxima possível, obtida quando se utiliza somente uma equipe; a sétima coluna contém as restrições ligadas às datas de início das atividades: a data condicionante na forma C<*data*>; o *buffer* imposto na forma <*atividade precedente (número) - buffer (em dias)*>; e a restrição de lógica vertical na forma <*atividade precedente (número) - pavimento (número)*> se a atividade for repetitiva e <*precedente (número)*> se a atividade for não repetitiva; e finalmente na sétima coluna, pode-se obter as despesas diretas de cada atividade.

PROBLEMA Nº 1

Tipo de Rede: Mista
Número de Pavimentos Repetitivos: 7

Tabela 1 - Dados de Tempo e de Disponibilidade de Recurso Monetário:

Período	N. de Dias	Rec. Disp.
1	23	60.794,158
2	20	52.864,485
3	21	56.829,504
4	22	72.817,803
5	22	155.988,072
6	20	39.096,616
7	23	134.922,576
8	21	77.301,688
9	21	10.211,336
10	23	9.574,744
11	19	11.983,458
12	21	14.149,292
13	23	11.701,691
14	20	11.029,670
15	21	35.650,277
16	22	91.638,008
17	22	59.338,569
18	20	51.920,266
19	23	67.518,114
20	21	63.805,705
21	21	83.073,581
22	23	92.481,525
23	19	49.438,303
24	21	13.441,850
25	23	11.282,006
26	20	9.810,440
27	21	22.996,230
28	22	5.761,484
29	22	2.206,786
30	20	13.253,193

Tabela 2 - Dados das Atividades

Nº	Denominação	Precedências
	SUPRA-ESTRUTURA	
1	Supra-Estrutura	0
	PAREDES E ESQUADRIAS	
	Paredes	
2	Alvenaria Externa	1
3	Alvenaria Interna	1
4	Encunhamento Alvenaria	2, 3
	Esquadrias	
5	Contramarcos de Alumínio	2
6	Caixilhos de Madeira	31
7	Portas de Madeira	6

Continuação...

8	Rodapés e Guarnições de Madeira	7
9	Esquadrias de Alumínio	26, 43
10	Portas Corta-fogo	18
	Vidros	
11	Vidros Comuns	9
12	Vidros Laminados	9, 34
13	Boxes e Espelhos	43
	COBERTURAS E IMPERMEAB.	
14	Regularização de Superfícies	57
15	Impermeabilização e Prot. Mecânica	14
	REVESTIMENTOS	
	Revestimentos Internos	
16	Chapisco Parede Interna	4, 55, 59
17	Emboço Parede Interna	15, 56, 60
18	Chapisco e Emboço da Escada	4
19	Requadro de Emboço Interno	5, 17
20	Reboco Parede Interna	19, 41, 61
21	Reboco da Escada	42
22	Azulejos	5, 41
23	Mármores em Paredes	5, 17
	Revestimentos Externos	
24	Chapisco Parede Externa	4
25	Emboço Parede Externa	24
26	Requadro de Emboço Externo	25
27	Pastilhas Cerâmicas	26
28	Granito na Fachada	24
	Revestimentos de Forros	
29	Chapisco de Tetos	2
30	Emboço de Tetos	29
31	Reboco de Teto	20
32	Forro de Gesso	31, 45
33	Forro de Madeira	27, 28
34	Peitoris de Sacadas	27
	Pintura	
35	Massa Corrida	30, 32, 46, 52, 58,
36	Enceramentos	8
37	Enceramentos sobre Forros	33
38	Pintura Latex	12, 35, 36, 63,
39	Pintura Esmalte em Portas	10
40	Pintura de Escadaria	39
	PISOS	
41	Contrapiso	17
42	Contrapiso da Escada	10
43	Pisos em Granito	22, 53
44	Pisos Cerâmicos	17, 22
45	Rejunte de Azulejos e Pisos Cerâmicos	51, 54
46	Piso Laminado	31
47	Piso em Granitina	21
48	Carpete	65
	INSTALAÇÕES	
	Aparelhos Hidro-Sanitários	
49	Metais e Acessórios	7, 54
50	Louças	43, 44
51	Tanques	22

Continuação...

52	Aquecedores	22
53	Banheiras	23
54	Tampos de Granito	50
	Inst. Hidráulico-sanitárias, Gás	
55	Prumadas de Água e Incêndio	1
56	Gás – Prumadas e Ramais	1
57	Hidráulica – Acabamentos ag/esg.	4
58	Acabamentos de Inc. e Gás	31
	Inst. Elétricas, Telefônicas	
59	Elétrica – Prumadas e Quadros	1
60	Elétrica – Tubulação de Parede	16
61	Elétrica – Cabos, Fiação e Disjuntores	17
62	Interruptores, Tomadas e Calefação	35
	Complementação da obra	
63	Fornecimento de Lareiras e Churrasqueiras	11
64	Paisagismo	15, 27
65	Limpeza	13, 37, 38, 40, 47, 49, 62, 64
66	*Dummy*	48

Tabela 2 (Continuação) – Dados das Atividades:

Nº	Tipo	Sentido	NminEq	NmaxEq	dlEq	Restrição	Despesa
1	R	A	4	20	395	-	261.679,20
2	R	A	4	17	202	B<1-21>	27.142,84
3	R	A	4	17	275	-	30.401,28
4	R	A	2	13	117	-	7.018,89
5	R	A	1	8	49	-	3.271,51
6	R	A	1	9	45	B<1-21>	3.526,10
7	R	A	1	12	85	-	29.006,36
8	R	A	1	15	135	-	39.060,94
9	R	A	2	17	238	-	63.775,57
10	R	A	1	3	5	-	2.563,40
11	R	A	1	6	37	-	10.831,75
12	R	A	1	4	16	-	5.170,24
13	R	A	1	8	48	-	18.031,20
14	R	A	2	10	51	-	15.166,29
15	R	A	3	10	80	-	18.429,08
16	R	A	2	14	126	-	12.349,25
17	R	A	4	20	739	-	47.197,42
18	R	A	1	12	56	-	3.614,42
19	R	A	3	9	78	-	3.683,92
20	R	A	2	10	76	-	5.627,02
21	R	A	1	4	9	-	598,32
22	R	A	1	10	77	-	9.308,42
23	R	A	1	7	47	-	44.943,98
24	R	A	2	9	49	-	4.544,04
25	R	A	3	19	336	-	20.192,55
26	R	A	3	12	80	-	3.816,30
27	R	A	3	22	324	-	36.384,75
28	R	A	1	4	9	-	14.640,48
29	R	A	1	9	60	-	3.889,84
30	R	A	1	14	133	-	8.575,35
31	R	A	1	5	18	-	1.002,79
32	R	A	1	8	39	-	5.591,74

Continuação...

33	R	A	1	7	21	-	4.593,76
34	R	A	1	9	72	C<385>	4.160,88
35	R	A	3	22	330	-	25.150,00
36	R	A	1	8	39	-	2.301,46
37	R	A	1	2	4	-	220.52
38	R	A	3	19	345	-	30.116,85
39	R	A	1	2	2	-	141.56
40	R	A	1	6	37	-	3.133,51
41	R	A	3	15	149	-	19.177,52
42	R	A	1	3	9	-	908,58
43	R	A	1	12	96	-	71.830,54
44	R	A	1	3	9	-	1.616,89
45	R	A	1	7	28	-	2.286,14
46	R	A	1	9	44	-	17.490,61
47	R	A	4	8	39	-	15.650,42
48	R	D	1	6	19	-	12.979,23
49	R	A	1	7	35	C<126>	94.831,38
50	R	A	1	8	40	-	36.958,09
51	R	A	1	4	7	-	3.196,64
52	R	A	1	3	9	-	17.780,67
53	R	A	1	2	4	-	3.295,79
54	R	A	1	4	12	-	14.210,20
55	R	A	2	4	7	-	58.499,78
56	R	A	1	4	7	-	5.214,55
57	R	A	1	4	7	-	10.476,56
58	R	A	1	4	7	-	6.544,63
59	R	A	1	4	7	-	27.231,02
60	R	A	1	8	33	-	17.662,11
61	R	A	1	7	28	-	15.724,51
62	R	A	1	3	5	-	26.114,26
63	NR	-	1	2	2	-	9.211,07
64	NR	-	1	1	1	LV<50..63-2>	2.171,95
65	R	D	1	7	26	-	647,55

Fonte:

MENDES JR., R., *Programação de Serviços em Edifícios Altos com Linha de Balanço: Uma Aplicação*. Orientação: HEINECK, L. F. M. Pós-Graduação em Engenharia de Produção e Sistemas. Universidade Federal de Santa Catarina, Florianópolis: 1995.

PROBLEMA Nº 2

Tipo de Rede: Mista
Número de Pavimentos Repetitivos: 12

Tabela 1 - Dados de Tempo e de Disponibilidade de Recurso Monetário:

Período	N. de Dias	Rec. Disp.
1	23	82.432,890
2	20	71.680,774
3	21	75.264,812
4	22	75.390,558
5	22	98.581,936
6	20	146.304,422
7	23	220.373,976
8	21	26.129,744
9	21	26.129,744
10	23	28.618,291
11	19	23.991,500
12	21	48.513,148
13	23	71.075,918
14	20	55.096,774
15	21	51.641,444
16	22	6.569,347
17	22	5.395,566
18	20	4.905,060
19	23	88.314,180
20	21	172.063,287
21	21	187.123,820
22	23	27.990,475
23	19	53.317,203
24	21	36.287,900
25	23	89.343,990
26	20	12.401,581
27	21	5.009,431

Tabela 2 - Dados das Atividades

Nº	Denominação	Precedências
1	Formas - Preparação e Colocação	0
2	Armaduras – Preparação	0
3	Armaduras – Colocação	1,2
4	Tub. e Cxs. Elétricas e Telefônicas nas Lajes	3
5	Tubulações de Gás nas Lajes	3
6	Caixas de Passagem para Tubulação Hidro-Sanitária	3
7	Concretagem	4,5,6
8	Desforma	7
9	Alvenaria, Vergas e Contravergas	8
10	Rasgos na Alvenaria para Tubulação de Incêndio	9
11	Rasgos na Alvenaria para Tubulação Hidráulica	9
12	Rasgos na Alvenaria para Tubulação Elétrica	9
13	Rasgos na Alvenaria para Tubulação de Esgoto	9
14	Rasgos na Alvenaria para Tubulação Telefônica	9

Continuação...

15	Rasgos na Alvenaria para Tubulação de Gás	9
16	Instalação da Tubulação de Incêndio	10
17	Instalação da Tubulação Hidráulica	11
18	Instalação da Tubulação Elétrica	12
19	Instalação da Tubulação de Esgoto	13
20	Instalação da Tubulação Telefônica	14
21	Instalação da Tubulação de Gás	15
22	Colocação dos Contramarcos	9
23	Fechamento dos Rasgos	16,17,18,19,20,21,22
24	Chapisco dos Forros	23
25	Chapisco das Paredes Internas	23
26	Chapisco das Paredes Externas	9
27	Emboço + Reboco dos Forros	24
28	Emboço + Reboco Interno	25,27
29	Emboço + Reboco Externo	26
30	Batentes e Esquadrias Metálicas	28,29
31	Peitoris	29
32	Pintura Externa	30,31
33	Regularização dos Pisos	30
34	Colocação dos Azulejos	33
35	Impermeabilização de Áreas Molhadas	33
36	Pisos Cerâmicos e Soleiras	35
37	Vidros	36
38	Colocação de Louças	36
39	Colocação de Portas	36
40	Fiação Elétrica, Telefônicas e Pontos	37,38,39
41	Forros	40
42	Rodapés e Guarnições	41
43	Pintura Interna das Paredes	42
44	Pintura de Portas e Marcos	42
45	Pintura dos Rodapés	42
46	Colocação de Espelhos Elétricos	43,44,45
47	Forração dos pisos	46
48	Assentamento dos Pisos Cerâmicos Hall/Escadaria	47
49	Colocação dos Corrimãos da Escadaria	48
50	Pintura Completa Hall/Escadaria	49
51	Colocação de Acessórios Elétricos Hall/Escadaria	50
52	Limpeza Final	34,51
53	*Dummy*	32,52

Tabela 2 (Continuação) – Dados das Atividades:

Nº	Tipo	Sentido	NminEq	NmaxEq	dlEq	Restrição	Despesa
1	R	A	8	20	925	-	315.458,00
2	R	A	4	20	328	-	70.895,00
3	R	A	4	9	82	-	70.895, 00
4	R	A	3	13	103	-	4.280,00
5	R	A	1	6	17	C<105>	2.200,00
6	R	A	1	4	7	-	1.500,00
7	R	A	5	11	66	-	198.180,00
8	R	A	2	10	69	-	100.400,00
9	R	A	7	20	815	B<7-21>	145.580,00
10	R	A	1	3	6	-	1.240,00
11	R	A	1	4	8	-	3.120,00

Continuação...

12	R	A	3	11	99	-	5.780,00
13	R	A	1	4	12	-	2.600,00
14	R	A	1	4	7	-	1.230,00
15	R	A	1	3	3	-	1.243, 00
16	R	A	1	6	19	-	1.350, 00
17	R	A	1	7	27	-	3.360,00
18	R	A	3	13	118	-	17.340,00
19	R	A	1	6	30	-	2.580,00
20	R	A	1	4	8	C<265>	1.200,00
21	R	A	1	6	19	-	1.200,00
22	R	A	1	7	34	-	15.125,00
23	R	A	3	11	90	-	11.414, 00
24	R	A	2	10	69	-	7.981,00
25	R	A	6	14	198	-	16.727, 00
26	R	A	5	9	55	-	5.628, 00
27	R	A	3	13	171	-	19.126, 00
28	R	D	6	20	496	-	20.356, 00
29	R	D	3	18	137	-	26.331, 00
30	R	D	4	12	109	LV<28,29-2>	115.750, 00
31	R	D	1	7	36	-	51.625, 00
32	R	D	3	15	137	-	12.261, 00
33	R	D	8	20	171	-	17.218, 00
34	R	D	8	13	901	-	54.067, 00
35	R	D	1	8	47	-	23.596, 00
36	R	D	4	15	148	-	12.800, 00
37	R	D	1	5	14	-	74.960, 00
38	R	D	1	8	33	-	39.152, 00
39	R	D	3	12	96	-	125.132,67
40	R	D	6	18	308	-	20.080,23
41	R	D	1	6	19	-	2.240,24
42	R	D	9	17	204	-	16.238,54
43	R	D	6	20	347	-	44.347,31
44	R	D	4	17	185	-	22.306,20
45	R	D	1	3	5	-	2.500,40
46	R	D	1	4	7	-	2.012,49
47	R	D	1	7	27	-	77.600,40
48	R	D	1	6	17	-	1843,98
49	R	D	1	4	7	-	1.213,00
50	R	D	1	4	12	-	1.534,75
51	R	D	1	1	1	-	1.313,23
52	R	D	4	15	137	LV<34,51-2>	7.156,33

Fonte:

OLIVEIRA, M. C. G., BRANDI, L. L., MAYA, J. G., *Aplicação da Linha de Balanço para Edifícios Altos: Um Estudo de Caso*. Orientação: HEINECK, L. F. M. Pós-Graduação em Engenharia de Produção e Sistemas. Universidade Federal de Santa Catarina, Florianópolis: 1996.

PROBLEMA Nº 3

Tipo de Rede: Mista
Número de Pavimentos Repetitivos: 11

Tabela 1 - Dados de Tempo e de Disponibilidade de Recurso Monetário:

Período	N. de Dias	Rec. Disp.
1	23	38.603,000
2	20	36.302,480
3	21	82.369,174
4	22	63.387,596
5	22	6.527,125
6	20	11.091,674
7	23	72.546,501
8	21	130.462,527
9	21	30.812,739
10	23	165.697,624
11	19	148.928,031
12	21	11.461,882
13	23	12.133,047
14	20	1.515,920

Tabela 2 - Dados das Atividades

Nº	Denominação	Precedências
1	Limpeza do Terreno	0
2	Locação da Obra	1
3	Escavação das Fundações	2
4	Formas da Fundação	3
5	Armadura	3
6	Concreto da Fundação	4, 5
7	Reaterro	6
8	Formas da Estrutura	7
9	Tubulação Elétrica das Lajes	8
10	Armadura da Estrutura e Laje	9
11	Concreto da Estrutura e Laje	10
12	Alvenaria	11
13	Cobertura	11
14	Rasgos para Tubulações de Incêndio	12
15	Rasgos para Tubulações Hidráulicas	12
16	Rasgos para Tubulações Elétricas	12
17	Rasgos para Tubulações de Esgoto	12
18	Rasgos para Tubulações Telefônicas	12
19	Rasgos para Tubulações de Gás	12
20	Instalação das Tubulações de Incêndio	14
21	Instalação das Tubulações Hidráulicas	15
22	Instalação das Tubulações Elétricas	16
23	Instalação das Tubulações de Esgoto	17
24	Instalação das Tubulações Telefônicas	18
25	Instalação das Tubulações de Gás	19
26	Fechamento de Rasgos	20, 21, 22, 23, 24, 25
27	Chapisco de Paredes Internas	26
28	Chapisco do Forro	26

Continuação...

29	Emboço das Paredes Internas	27
30	Emboço do Forro	28
31	Chapisco + Emboço das Paredes Externas	26
32	Colocação de Batentes e Esquadrias	30, 31
33	Colocação de Azulejos	32
34	Regularização do Piso Interno	33
35	Piso Cerâmico	34
36	Rodapé de Madeira	34
37	Esquadrias	35, 36
38	Fiação Elétrica	37
39	Forro	37
40	Colocação de Aparelhos	37
41	Pintura Interna	38, 39
42	Revestimentos Externos	31
43	Colocação de Metais	40
44	Tomadas Internas, Espelhos e Interruptores	41
45	Limpeza Geral	13, 29, 40, 44
46	*Dummy*	42, 45

Tabela 2 (Continuação) – Dados das Atividades:

Nº	Tipo	Sentido	NminEq	NmaxEq	dlEq	Restrição	Despesa
1	NR	-	6	9	36	-	1.798,00
2	NR	-	1	3	6	-	2.180,00
3	NR	-	1	2	2	-	1.830,00
4	NR	-	10	13	129	-	18.713,00
5	NR	-	10	13	118	-	14.082,00
6	NR	-	18	20	20	-	14.202,00
7	NR	-	1	5	8	-	690,00
8	R	A	12	21	298	-	31.486,00
9	R	A	9	11	89	-	3.780,00
10	R	A	13	15	135	-	22.575,00
11	R	A	4	6	24	-	23.897,00
12	R	A	14	20	670	B<11-21>	14.241,00
13	NR	-	13	20	336	-	81.869,00
14	R	D	1	3	5	-	336,00
15	R	D	1	4	7	-	1.006,00
16	R	D	6	11	89	-	5.241,00
17	R	D	1	4	11	-	756,00
18	R	D	1	3	6	-	336,00
19	R	D	1	3	3	-	336,00
20	R	D	1	4	16	C<130>	840,00
21	R	D	4	6	24	-	2.520,00
22	R	D	12	12	107	-	13.104,00
23	R	D	5	8	56	-	1.890,00
24	R	D	1	4	7	-	840,00
25	R	D	1	4	16	C<130>	840,00
26	R	D	5	12	82	-	13.356,00
27	R	D	12	16	191	-	15.237,00
28	R	D	5	9	62	-	6.065,00
29	R	D	12	20	476	-	16.409,00
30	R	D	6	15	155	-	14.535,00
31	R	D	6	13	132	-	13.296,00
32	R	D	5	13	115	-	102.253,00

Continuação...

33	R	D	11	20	527	-	26.564,00
34	R	D	6	14	155	-	13.085,00
35	R	D	4	13	104	-	7.840,00
36	R	D	9	17	224	-	14.667,00
37	R	D	12	19	287	-	102.209,00
38	R	D	6	9	53	-	28.828,00
39	R	D	1	4	8	-	772,00
40	R	D	6	12	84	-	71.959,00
41	R	D	18	20	919	-	27.836,00
42	R	D	1	6	37	-	4.480,00
43	R	D	1	8	32	-	65.961,00
44	R	D	6	11	79	-	2.335,00
45	R	D	5	14	124	LV<29..44-2>	5.414,00

Fonte:

TRISTÃO, M. D. T., FERREIRA, J. C. G., ANDRADE, V. A. *Utilização da Técnica de Linha de Balanço Para Planejamento, Programação e Controle de um Conjunto Residencial de Blocos de Edifícios: Estudo de Caso.* Orientação: HEINECK, L. F. M. Pós-Graduação em Engenharia de Produção e Sistemas. Universidade Federal de Santa Catarina, Florianópolis: 1995.

PROBLEMA Nº 4

Tipo de Rede: Mista

Número de Pavimentos Repetitivos: 12

Tabela 1 - Dados de Tempo e de Disponibilidade de Recurso Monetário:

Período	N. de Dias	Rec. Disp.
1	23	118.201,735
2	20	128.718,716
3	21	249.661,548
4	22	30.205,714
5	22	31.039,353
6	20	29.683,959
7	23	40.881,837
8	21	41.851,775
9	21	58.984,958
10	23	36.604,553
11	19	16.361,578
12	21	116.586,162
13	23	22.546,987
14	20	10.321,011
15	21	22.698,761
16	22	46.807,844
17	22	42.739,661
18	20	36.250,847
19	23	40.940,000
20	21	98.841,372
21	21	92.647,401
22	23	60.581,145
23	19	4.184,722
24	21	4.625,219
25	23	5.065,716
26	20	5.647,323
27	21	206.377,356
28	22	192.518,761
29	22	9.043,848
30	20	8.704,525
31	23	16.918,654
32	20	14.711,873
33	21	15.447,467
34	22	16.183,060
35	22	18.191,340
36	20	14.991,831
37	23	1.159,382

Tabela 2 - Dados das Atividades

Nº	Denominação	Precedências
1	Formas	0
2	Armadura da Laje	1
3	Tubulação Elétrica da Laje	2
4	Concretagem	3

Continuação...

5	Desforma	4
6	Alvenaria	5
7	Instalações de Incêndio	6
8	Instalações Elétricas	6
9	Instalações Telefônicas	6
10	Instalações Hidro-Sanitárias	6
11	Chapisco Interno	7,8,9,10
12	Marcos/Contramarcos	11
13	Chapisco Externo	11
14	Contrapiso	12
15	Impermeabilização	14
16	Revestimento do Forro	14
17	Emboço Interno	15,16
18	Reboco Interno	17
19	Colocação de Azulejos	17
20	Forro de Gesso	19
21	Piso Cerâmico	18,20
22	Reboco Externo (massa única)	13
23	Esquadrias	21, 22
24	Louças/Metais	23
25	Revestimento Externo (Fulget)	23
26	Vidros	24
27	Pintura Interna	26
28	Passagem da Fiação	27
29	Tomadas/Interruptores	28
30	Revestimento em Carpete	29
31	Rodapé	30
32	Limpeza Final	25,31
33	*Dummy*	32

Tabela 2 (Continuação) - Dados das Atividades:

Nº	Tipo	Sentido	NminEq	NmaxEq	dlEq	Restrição	Despesa
1	R	A	10	20	492	-	211.166,40
2	R	A	18	20	692	-	82693,44
3	R	A	3	9	53	-	1.584,00
4	R	A	12	13	157	-	112.579,20
5	R	A	10	18	201	-	98.400,00
6	R	A	9	20	1064	B<4-1>	90.480,00
7	R	A	11	20	1318	-	77.292,00
8	R	A	4	9	82	-	5.040,00
9	R	A	7	20	493	C<125>	26.028,00
10	R	A	6	14	153	-	5.040,00
11	R	A	5	18	216	-	12.324,00
12	R	A	5	18	214	-	14.112,00
13	R	A	4	17	256	-	24.930,00
14	R	A	3	12	85	-	7.287,00
15	R	A	4	19	246	-	17.280,00
16	R	D	9	20	646	-	13.272,00
17	R	D	7	20	448	-	11.037,60
18	R	D	7	20	453	-	16.343,04
19	R	D	1	6	19	-	1.940,40
20	R	D	3	16	177	-	9.600,00
21	R	D	7	20	818	-	208.260,00

Continuação...

22	R	D	2	11	66	-	117.120,00
23	R	D	1	7	43	C<415>	27.000,00
24	R	D	9	20	1523	-	37.222,00
25	R	D	9	20	440	-	170.016,00
26	R	D	2	11	101	-	3.960,00
27	R	D	1	8	38	-	363.000,00
28	R	D	4	14	156	-	7.296,00
29	R	D	3	12	121	-	12.381,00
30	R	D	4	20	754	-	57.928,00
31	R	D	3	20	404	-	57.928,00
32	R	D	9	20	312	LV<25,31-2>	6.390,00

Fonte:

GUCH, D. U., BRANDÃO, D. Q., PAZ, M. A. S., *Aplicação da Linha de Balanço NA Programação de Obras de Construção Civil: Estudo para o Caso de Edifícios Altos.* Orientação: HEINECK, L. F. M. Pós-Graduação em Engenharia de Produção e Sistemas. Universidade Federal de Santa Catarina, Florianópolis: 1995.

PROBLEMA Nº 5

O problema 5 possui os mesmos dados do problema 3, com exceção dos dados referentes à tabela 1:

Tabela 1 - Dados de Tempo e de Disponibilidade de Recurso Monetário:

Período	N. de Dias	Rec. Disp.
1	23	37.163,538
2	20	38.578,402
3	21	81.532,713
4	22	63.624,946
5	22	6.962,267
6	20	19.327,609
7	23	129.941,882
8	21	85.387,356
9	21	110.709,822
10	23	214.469,015
11	19	10.939,489
12	21	12.949,627

PROBLEMA Nº 6

Fonte: Problema Nº 3.

Número de Pavimentos Repetitivos: 11

Tabela 1 - Dados de Tempo e de Disponibilidade de Recurso Monetário:

Período	N. de Dias	Rec. Disp.
1	23	233.733,451
2	20	89,985,550
3	21	15.449,439
4	22	6.117,561

Tabela 2 - Dados das Atividades

Nº	**Denominação**	**Precedências**
34	Regularização do Piso Interno	0
35	Piso Cerâmico	34
36	Rodapé de Madeira	34
37	Esquadrias	35, 36
38	Fiação Elétrica	37
39	Forro	37
40	Colocação de Aparelhos	37
41	Pintura Interna	38, 39
42	Revestimentos Externos	0
43	Colocação de Metais	40
44	Tomadas Internas, Espelhos e Interruptores	41
45	Limpeza Geral	43, 44
46	*Dummy*	42, 45

Tabela 2 (Continuação) - Dados das Atividades:

Nº	Tipo	Sentido	NminEq	NMaxEq	dlEq	Restrição	Despesa
34	R	D	6	14	155	-	13.085,00
35	R	D	4	13	104	-	7.840,00
36	R	D	9	17	224	-	14.667,00
37	R	D	12	19	287	-	102.209,00
38	R	D	5	9	53	-	28.828,00
39	R	D	1	4	8	-	772,00
40	R	D	6	12	84	-	71.959,00
41	R	D	18	20	919	-	27.836,00
42	R	D	1	6	37	C<25>	4.480,00
43	R	D	1	8	32	-	65.961,00
44	R	D	5	11	79	B<41-1>	2.335,00
45	R	D	5	14	124	LV<10,11-1>	5.414,00

PROBLEMA Nº 7

Fonte: Problema Nº 3.

Número de Pavimentos Repetitivos: 11

Tabela 1 - Dados de Tempo e de Disponibilidade de Recurso Monetário:

Período	N. de Dias	Rec. Disp.
1	23	118.625,091
2	20	78.609,802
4	21	10.350,107

Tabela 2 - Dados das Atividades

Nº	Denominação	Precedências
38	Fiação Elétrica	0
39	Forro	0
40	Colocação de Aparelhos	0
41	Pintura Interna	38, 39
42	Revestimentos Externos	0
43	Colocação de Metais	40
44	Tomadas Internas, Espelhos e Interruptores	41
45	Limpeza Geral	43, 44
46	*Dummy*	42, 45

Tabela 2 (Continuação) - Dados das Atividades:

Nº	Tipo	Sentido	NMinEq	NMaxEq	D1Eq	Restrição	Despesa
38	R	D	5	9	53		28.828,00
39	R	D	1	4	8		772,00
40	R	D	6	12	84		71.959,00
41	R	D	18	20	919		27.836,00
42	R	D	1	6	37	C<23>	4.480,00
43	R	D	1	8	32		65.961,00
44	R	D	5	11	79	B<41-1>	2.335,00
45	R	D	5	14	124	LV<10,11-1>	5.414,00

PROBLEMA Nº 8

Fonte: Problema Nº 4.

Número de Pavimentos Repetitivos: 12

Tabela 1 - Dados de Tempo e de Disponibilidade de Recurso Monetário:

Período	N. de Dias	Rec. Disp.
1	23	7.317,145
2	20	6.362,735
3	21	6.6.999,009
4	22	6.999,009
5	22	6.999,009
6	20	178.770,599
7	23	205.726,931
8	21	17.605,266
9	21	13.679,583
10	23	29.315,415
11	19	29413,220
12	21	35.253,381
13	23	1.981,836

Tabela 2 - Dados das Atividades

Nº	Denominação	Precedências
24	Louças/Metais	0
26	Vidros	24
27	Pintura Interna	26
28	Passagem da Fiação	27
29	Tomadas/Interruptores	28
30	Revestimento em Carpete	29
31	Rodapé	30
32	Limpeza Final	31
33	*Dummy*	32

Tabela 2 (Continuação) - Dados das Atividades:

Nº	Tipo	Sentido	NMinEq	NMaxEq	D1Eq	Restrição	Despesa
24	R	D	9	20	1523	-	37.222,00
26	R	D	2	11	101	C<100>	3.960,00
27	R	D	1	8	38	-	363.000,00
28	R	D	4	14	156	-	7.296,00
29	R	D	3	12	121	C<140>	12.381,00
30	R	D	4	20	754	B<29-1>	57.928,00
31	R	D	3	20	404	-	57.928,00
32	R	D	9	20	312	LV<31-1>	6.390,00