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

UM ALGORITMO EVOLUTIVO PARA A PROGRAMAÇÃO DE PROJETOS MULTI-MODOS COM NIVELAMENTO DE RECURSOS LIMITADOS

Tese Submetida À Universidade Federal De Santa Catarina Para A Obtenção
Do Título De Doutor Em Engenharia De Produção.

Oscar Ciro López Vaca

Florianópolis, Março de 1995

UM ALGORITMO EVOLUTIVO PARA A PROGRAMAÇÃO DE PROJETOS MULTI-MODOS COM NIVELAMENTO DE RECURSOS LIMITADOS

Oscar Ciro López Vaca

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.

Prof. Osmar Possamai, Dr.
Coordenador do Curso

BANCA EXAMINADORA:

Prof. Ricardo Miranda Barcia, Ph.D.
Orientador

Prof. Osama Eyada, Ph.D.
Examinador Externo

Prof. Carlos Augusto de Alcantara Gomes, D.Sc.
Examinador Externo

Prof. Luiz Fernando M. Heinecke, Ph.D.

Prof. Fernando Álvaro Ostuni Gauthier, Dr.

Prof. Rogério Cid Bastos, Dr.
Moderador

Com amor para Sandra, Eduardo, Jacqueline e Renato

AGRADECIMENTOS

É muito difícil agradecer a todas as pessoas que contribuíram com amor, amizade, companheirismo, abnegação e trabalho para a minha formação, sem correr o risco de cometer alguma injustiça. Gostaria de agradecer profundamente às seguintes pessoas:

- Ao professor Ricardo M. Barcia, por me incentivar a fazer o doutorado e pela orientação.
- Ao professor Osama Eyada, pela paciência com o meu Inglês e pela orientação.
- Aos professores Adedeji B. Badiru, diretor do Expert Systems Laboratory, School of Industrial Engineeering, University of Oklahoma, Norman OK, USA. e Raymond Li, do Department of Bussiness Systems of the Monash University, Clayton, Australia, por terem fornecido o material necessário à execução dos experimentos.
- Ao professor Luiz F. Heineck, pela ajuda e sugestões.
- Ao CNPq, pelo suporte financeiro.
- Ao André e a Carla, pela amizade.
- Ao José e a Kristini, por seus trabalhos de revisão do manuscrito.
- Aos colegas e funcionários do departamento, pela amizade e o estímulo.
- Ao Ciro e a Aida, pelo amor, o apoio material, a renúncia e principalmente, a Deus por ter-los escolhido como meus pais.
- A minha família na Bolívia, que mesmo a distância me brindou amor e apoio, e a toda minha família brasileira, por ter me acolhido como filho e irmão.
- Finalmente, um especial reconhecimento a minha esposa Sandra pelo amor, a confiança, a abnegação, a compreensão e o apoio em todas as horas, e aos meus filhos Eduardo, Jacqueline e Renato, por entender, suportar e perdoar as minhas falhas.

SUMÁRIO

CAPÍTULO 3

CAPÍTULO 4

CAPÍTULO 5

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Um projeto de construção é composto de um conjunto de tarefas ou atividades que são executadas segundo uma ordem previamente determinada, exigindo tempo e recursos na realização dos objetivos da administração. É muito comum que estas atividades devam ser programadas sob recursos limitados. Nestas condições, frequentemente é necessário tomar decisões sobre a ordem de execução das tarefas, de modo a minimizar o aumento na duração do projeto além da duração calculada sem restrição de recursos. Por outro lado, há situações em que o custo da flutuação dos recursos empregados pelas atividades pode ser substancial.

Os modelos desenvolvidos para a tratar o problema da alocação, não abordam a programação sob restrição de recursos e o problema do nivelamento num único procedimento. A principal razão é a natureza antagônica das premissas sobre as quais elas se fundamentam. Enquanto que no problema de restrição de recursos é permitido estender a duração do projeto, no processo de nivelamento assume-se que o projeto tem uma data de realização fixa e recursos suficientes.

Devido à sua natureza combinatorial, o problema de programação de projetos com restrição de recursos, é considerado um dos problemas mais desafiantes. A busca de soluções através de abordagens analíticas tem alcançado relativo sucesso somente em problemas de pequeno porte. Desta forma, esforços vem sendo realizados na procura de soluções mais robustas através de abordagens heurísticas.

Este trabalho apresenta um modelo que utiliza os Algoritmos Genéticos como abordagem heurística para resolver o problema de alocação de recursos limitados em ambiente de múltiplos modos. Além disto, o conceito de nivelamento de recursos é incorporado ao procedimento como um mecanismo guia na busca de uma solução de mínima duração, com padrões de flutuação dos recursos utilizados de alguma forma reduzidos.

O modelo é implementado em C++ e desenvolvido para ambientes de computador pessoal compatível com o padrão IBM. O seu desempenho computacional é testado usando 25 problemas extraído da literatura especializada. Os resultados obtidos da comparação do modelo com os resultados reportados em dois estudos relacionados com o tema, indicam que a abordagem desenvolvida fornece soluções melhores, robustas e flexíveis com esforço computacional razoável.

ABSTRACT

Planning in the construction industry refers to a set of activities to be undertaken according to a predefined sequence, consuming time and resources in order to achieve management objectives. Frequently, activities must be performed under limited resources conditions. In addition, activities are executed in parallel and resources are not always sufficient to satisfy the demands of concurrent activities. The consequence is an increase in the project duration when compared with a non-constrained situation. Also, there are many instances where short-term variations in labor, equipment or money have a negative effect in the overall project cost.

Current models for the resource-constrained project scheduling problem do not incorporate resource leveling in the same approach. All approaches to either resource-constrained or resource leveling problems are developed under antagonic assumptions, such as limited and unlimited resource availabilities respectively. Furthermore, while extending the total project duration is acceptable in resource-constrained problems, it is not acceptable for the resource leveling problem.

Most varieties of this problem are of the NP-hard combinatorial type, making the generation of the optimal solution using purely mathematical approaches computationally impractical even for the simplest real-world sized projects. In order to overcome the practical limitations of mathematical-based approaches, heuristic-based solutions have been extensively developed.

This study is concerned with the problem of project scheduling under constrained resources and multiple performing modes. A Genetic Algorithms (GA) procedure is proposed as a heuristic solution to the problem. Furthermore, the resource leveling concept is incorporated to the procedure as a guidance mechanism in the search for a solution with not only the shortest possible duration, but with the best leveling compromise as well. The model implementation is based on the object-oriented paradigm and coded in the C++ language.

The model's computational performance was tested using a set of 25 example projects taken from the literature. Additionally, the model was faced against two benchmarks available in the literature. The results obtained indicate that the GA-based approach provides better solutions to the problem, while keeping the computational effort within reasonable limits. In addition, the GA-based approach is more flexible and provides robust solutions in the domain of construction scheduling problem.

CAPÍTULO 1
INTRODUÇÃO

1.1 Origem Do Trabalho

Um projeto de construção é composto de um conjunto de tarefas ou atividades que são executadas segundo uma ordem previamente determinada, exigindo tempo e recursos para serem realizadas, visando satisfazer os objetivos da administração.

De modo a satisfazer os objetivos estabelecidos, torna-se necessário fazer um planejamento e uma programação adequada, bem como o controle pertinente durante a execução das operações.

Na literatura da área, os termos *planejamento* e *programação* são objetos de confusão e, frequentemente, usados como sinônimos. O planejamento na sua concepção mais ampla pode ser entendido como a "*função administrativa que compreende a seleção e estudo de objetivos, diretrizes, planos, processos e programas, realizados dentro de um enfoque sistêmico, na construção de um todo equilibrado*" (Assed, 1986)

O planejamento, dentro do processo de administração de projetos e ao nível de produção é definido como a enumeração das atividades associadas com o projeto e a determinação da ordem na qual elas devem ocorrer. A programação é entendida como o escalonamento destas atividades com respeito à duração e aos recursos necessários à execução de cada uma delas.

A programação de um projeto é uma tarefa bastante difícil devido, principalmente, à grande variação na composição dos recursos envolvidos, que provoca um problema combinatorial elevado. Em geral, os métodos conhecidos para encontrar uma solução exata para este tipo de problemas, abrange um espaço de busca da solução que cresce exponencialmente com o número de variáveis envolvidas, por esta razão classificados como problemas NP.

Devido à sua natureza combinatorial, o problema de programação de projetos com restrição de recursos, é considerado um dos problemas mais desafiantes. A literatura apresenta, virtualmente, centenas de trabalhos e revisões bibliográficas relacionadas com a questão. A busca de soluções através de técnicas de programação matemática tem alcançado relativo sucesso somente em problemas de pequeno porte.

Neste contexto, esforços vem sendo realizados na procura de soluções mais robustas através de abordagens heurísticas. Para Davis e Patterson (1975), os procedimentos baseados em heurísticas para a programação de projetos sob restrição de recursos, são os meios práticos mais adequados de obter soluções viáveis para problemas de grande complexidade, do tipo geralmente encontrado na área.

Neste sentido, abordagens baseadas em heurísticas de busca local, como por exemplo Simulação por Anelamento, Algoritmos Genéticos e Tabu Search, são formas alternativas de procura do espaço de solução fundamentadas no Princípio da Vizinhança e objetos de recentes pesquisas para o problema de programação em geral.

1.2 Justificativa Do Trabalho

O CPM (Criticai Path Method) e o PERT (Program Evaluation and Review Technique) entre várias outras técnicas similares, são as ferramentas mais comumente utilizadas, desde a sua criação nos anos 50, para o planejamento e a programação de projetos. Entretanto, é reconhecido que as técnicas básicas empregadas no PERT e CPM são limitadas no sentido de não considerar limites na disponibilidade dos recursos.

É muito comum que as atividades de um projeto devam ser programadas sob recursos limitados. Nestas condições, frequentemente é necessário tomar decisões sobre a ordem de execução das tarefas, de modo a minimizar o aumento na duração do projeto além da duração sem restrição de recursos originalmente calculada usando o procedimento PERT/CPM padrão.

Como resultado, vários trabalhos tem investigado diferentes abordagens que levam em consideração a limitação dos recursos. Porém, o problema de ordenar um conjunto de atividades de modo a que nenhuma relação de precedência seja quebrada e, ao mesmo tempo, satisfazer os limites impostos aos recursos, não é uma tarefa fácil. A dificuldade aumenta se, simultaneamente, algum critério de otimização é introduzido, como por exemplo: minimizar a duração ou minimizar o custo de um dado projeto. Em geral, o objetivo mais comum da programação de atividades é encontrar uma ordenação viável que minimize a duração do projeto. Por outro lado, a programação de projetos sob restrição de recursos é um tipo de problema representativo da classe de problemas combinatoriais NP-completos (Wiest 1967, Lenstra e Rinnooy Kan 1978, Blazewicz *et al.* 1983, Boctor 1990).

Contudo, obter programações viáveis ou ótimas, para a maioria dos problema práticos, permanece em princípio, computacionalmente impraticável através de abordagens analíticas, como por exemplo: programação matemática (Wiest 1967, Davis e Patterson 1975, Patterson 1984, Moder *et al.* 1983, Davis et. al. 1992, Khattab e Choobineh 1991).

Deste modo, procedimentos baseados em heurísticas tem sido extensivamente utilizados. As abordagens heurísticas são hoje os meios mais eficientes para obter *boas* soluções em tempo computacional razoável (Davis e Patterson 1975, Moder *et al.* 1983, Boctor 1990, Khattab e Choobineh 1991).

Por outro lado, há situações em que o custo da flutuação dos recursos empregados pelas atividades pode ser substancial. Desta forma, o problema de nivelamento surge quando

existem recursos suficientemente disponíveis e deseja-se reduzir a variação dos seus padrões de utilização ao longo de um projeto que tem uma data determinada de término.

Entretanto, os modelos desenvolvidos para tratar o problema da alocação, não consideram a possibilidade de tratar a questão da programação sob restrição de recursos e o nivelamento destes ao mesmo tempo, ou seja, ambos incorporados num único procedimento. A principal razão desta deficiência é a natureza antagônica das premissas sobre as quais eles se fundamentam. Enquanto que no problema de restrição de recursos é permitido estender a duração do projeto, isto não acontece na questão do nivelamento. Além do mais, o processo de nivelamento de recursos é realizado sob a hipótese de que basicamente não existe limite na utilização dos meios, contrariamente ao problema de alocação de recursos restritos.

Neste sentido, é desenvolvido um modelo que utiliza os Algoritmos Genéticos como abordagem heurística, objetivando, além de preencher a deficiência acima apontada, resolver o problema de alocação de recursos em ambiente de múltiplos modos. Geralmente, é possível executar uma atividade de várias maneiras, cada uma delas utilizando uma combinação de recursos-durações diferente. A despeito da maioria dos modelos tradicionais de programação de projetos sob restrição de recursos, onde cada atividade é realizada em, exatamente, um único modo, o modelo desenvolvido incorpora vários modos alternativos para a execução de cada uma das atividades.

Em resumo, o modelo incorpora características que permitem resolver o problema de programação de projetos sujeito à restrição de múltiplos recursos e múltiplos modos, levando em consideração a flutuação na utilização de recursos de pesos diferentes.

1.3 Objetivos Do Trabalho

O trabalho desenvolvido tem dois objetivos principais:

1. desenvolver uma abordagem heurística baseada em Algoritmos Genéticos para tratar o problema de programação de projetos com restrição e nivelamento de múltiplos recursos em ambiente de múltiplos modos,
2. identificar as medidas de desempenho do modelo, comparando a heurística desenvolvida com alguns procedimentos existentes.

Paralelamente, a pesquisa propõe-se a:

1. apresentar uma classificação estruturada dos diferentes problemas de programação,
2. implementar computacionalmente o modelo proposto e, finalmente,

3. delinear a viabilidade de aplicação do procedimento a outras áreas onde o domínio do problema tenha características similares.

1.4 Organização Do Trabalho

O presente trabalho se constitui de cinco capítulos, mais a bibliografia empregada e referênciada e os anexos.

O Capítulo 1 introduz o tema, abrangendo: considerações iniciais, justificativas, objetivos e organização do trabalho.

No Capítulo 2 é apresentada a revisão do estado da arte do problema de programação em geral. É proposta uma classificação dos diversos problemas de programação, bem como são formuladas as premissas básicas que fundamentam o presente trabalho. O capítulo dedica a sua última parte a uma revisão sucinta dos fundamentos teóricos dos algoritmos genéticos.

O Capítulo 3 introduz o desenvolvimento do modelo heurístico proposto para resolver o problema em questão, nas bases formuladas no capítulo 2. O capítulo descreve em detalhes os fundamentos da heurística implementada e delineia a implementação computacional e o planejamento dos experimentos que serviram como base de validação da proposta.

O Capítulo 4 é dedicado à apresentação dos resultados numéricos decorrentes da aplicação da versão computacional do modelo desenvolvido.

O Capítulo 5 apresenta as principais conclusões e recomendações para futuros trabalhos, baseado, fundamentalmente na avaliação dos resultados apresentados no quarto capítulo.

Finalmente, é apresentada a bibliografia citada e usada ao longo do trabalho, bem como são anexados os problemas empregados na execução dos experimentos de validação do modelo.

CAPÍTULO 2

REVISÃO DA LITERATURA

2.1 O Problema de Programação

O problema de programação pode ser definido, de um modo geral, como a alocação de recursos no tempo de forma a executar um conjunto de tarefas (Maccarthy and Liu, 1993). Este conceito é de vital importância para várias atividades industriais, particularmente nos ambientes de manufatura e de projetos de construção.

A maioria dos problemas de programação envolvem diversos fatores, tais como a priorização de tarefas, requisitos de data de realização, restrições de custos, precedência de operações, demanda, disponibilidade e capacidade de recursos, etc. Por tudo isto, é bem sabido que, frequentemente, muitos destes problemas não são facilmente resolvidos, e que a determinação da melhor solução requer a solução de um problema de otimização combinatorial. Qualquer abordagem que busque uma solução para um determinado problema de programação é, na realidade, uma forma diferente de formular e resolver um problema de otimização. Desta forma, diferentes problemas de programação levam, naturalmente, a modelos diferentes. Assim, uma grande variedade de resultados existe para um vasto conjunto de problemas.

Neste capítulo serão discutidos alguns dos problemas fundamentais da teoria de programação e alguns dos métodos fundamentais e/ou utilizados na sua resolução. Para os propósitos deste trabalho, não haverá necessidade de definições matematicamente rigorosas da maioria dos conceitos envolvidos no problema de programação, na otimização combinatorial e na teoria de complexidade.

2.2 Revisão Da História Do Desenvolvimento Da Teoria Da Programação

O campo da teoria de programação é muito dinâmico e reconhecidamente, começou no início dos anos 50 com o trabalho pioneiro de Johnson (1954) sobre programação flowshop de duas máquinas. Este trabalho, junto com os trabalhos de Jackson e Smith, formam a base da teoria clássica do sequenciamento de máquinas. (MacCarthy e Liu, 1993). Desde então, a literatura tem crescido explosivamente. Vários livros tem sido publicados sobre o tema, entre os quais, Baker (1974), Coffman (1976), Lenstra (1977), Bellman *et al.* (1982), French (1986), Ibaraki (1987).

Por outro lado, as técnicas padrões da administração de projetos, como o PERT (Program Evaluation Review) e o CPM (Critical Path Method), vem sendo largamente

empregadas desde meados da década de 50. Este métodos objetivam principalmente, minimizar a duração do projeto, assumindo que recursos de vários tipos necessários a sua execução, estão suficientemente disponíveis. Entretanto, na prática, estes recursos são disponíveis em quantidades limitadas. Como resultado, tem havido uma atenção crescente para o problema de alocação de recursos associado com o planejamento e a programação de projetos. Boas referências sobre estas técnicas básicas são Moodier and Phillips (1968), Lockyer, K.G. (1981); Callahan, M. T *et al.* (1992).

Embora os problemas de programação da produção e os problemas de programação de projetos tenham sido abordados através de modelos próprios, as similaridades conceituais entre ambos tem estimulado interesses de aplicabilidade recíproca dos métodos de solução (Davis, 1973).

2.3 O Problema De Programação No Contexto Dos Problemas De Otimização Combinatorial

A otimização combinatorial é um dos campos da programação matemática, cuja aplicação é encontrada numa ampla gama de áreas, tais como, engenharia, economia, ciências sociais e outras. Um problema de otimização combinatorial intenta encontrar uma solução, maximizando (ou minimizando) uma função objetivo definida sobre um conjunto combinatorial (ou discreto) (Ibaraki, 1988).

Segundo Ibaraki, um problema de otimização é geralmente definido como:

P: Maximizar (minimizar) $f(x)$ (1.1)

sujeito a $x \in S$,

ou por conveniência de notação, como

$$P : \underset{x \in S}{Max}\ f(x) \tag{1.2}$$

ou

P: Max $\{f(x) \mid x \in S\}$ (1.3)

onde $\boldsymbol{S} \subset \boldsymbol{X}$ denota a *região viável* no espaço $\boldsymbol{X}$. Em outras palavras, $\boldsymbol{S}$ é o conjunto de *soluções viáveis* que satisfaz as restrições impostas. A função $f: S \rightarrow R$, onde $\boldsymbol{R}$ é o conjunto dos números reais, é chamada *função objetivo*. Uma solução viável $x \in \boldsymbol{S}$ é *ótima* se nenhuma outra solução viável satisfaz $f(y)>f(x)$ para o caso de maximização ou $(f(y)<f(x)$ no caso de minimização.

Um problema de otimização ***P*** é denominado de *problema combinatorial de otimização* se ***X*** e ***S*** são de alguma forma combinatoriais ou discretos. ***X*** e/ou ***S*** é *combinatorial* se eles são conjuntos discretos de elementos finitos ou elementos infinitos contáveis.

De acordo com Ibaraki, são considerados como conjuntos combinatoriais típicos o *conjunto dos inteiros* ***Z***, o conjuntos dos *n-vetores de inteiros* Z^n, o conjunto de políticas Σ^* geradas a partir de um conjunto finito de decisões Σ, qualquer *conjunto finito* e seus *sub-conjuntos* tais como o *conjunto de vetores 0-1 n-dimensional*, contendo 2^n elementos, o conjunto de *permutações de n objetos*, contendo *n!* elementos, e a família de *todos os sub-conjuntos de um conjunto finito de n-objetos*, contendo 2^n elementos. Um *grafo* de vértices e arcos finitos é outro exemplo de um objeto combinatorial.

Entre os exemplos típicos de otimização, Ibaraki menciona os problemas relacionados com grafos e redes: *maximum matching* problem, *maximum clique* problem, *maximum vertex packing* problem, *minimum vertex cover* problem, mínima cobertura de arestas, *problema do caminho mínimo*, *problema de fluxo máximo*, expansão de árvore de custo mínimo, expansão de árvore de custo mínimo direcionado, *problema de atribuição* e o *problema do caixeiro viajante*; problemas relativos à seleção de sub-conjuntos ótimos de conjuntos finitos: cobertura de conjuntos, partição de conjuntos, *set packing* problem; e problemas relativos à localização ótima de *n* objetos, denominado *problemas de localização:* problema dos p-centros, problema das p-medianas e *plant location problem*.

Para Ibaraki, a Programação Linear (PL) e a Programação Inteira (PI) e seus casos especiais como a PI pura, problema PI misto, problema 0-1, problema 0-1 misto, problema da mochila, problema da mochila 0-1 e problema da mochila multi-restrito, são também importantes membros do conjunto dos problemas de otimização combinatorial.

De acordo com este autor, a *programação* ou a *teoria de sequenciamento* é outra fonte importante de problemas de otimização. Entre estes podem ser citados, o problema de sequenciamento de uma máquina, programação de máquinas em paralelo, programação de múltiplos processadores, *bin packing problem*, o problema da programação flow-shop e job-shop. A Figura 2.1 mostra a classificação dos problemas combinatoriais segundo Ibaraki.

Por outro lado, Müller-Merbach (1981), divide os problemas combinatoriais em três tipos puros: (1) *problemas de atribuição,* onde há dois (ou mais) conjuntos de objetos discretos com os quais pares de elementos unitários de cada conjunto devem ser formados, (2) o *problema de sequenciamento,* onde existe um conjunto de objetos discretos (ou um sub-conjunto deles) que tem que ser ordenado em sequência e, (3) o *problema de seleção* onde, dado um conjunto de objetos discretos um sub-conjunto deve ser escolhido.

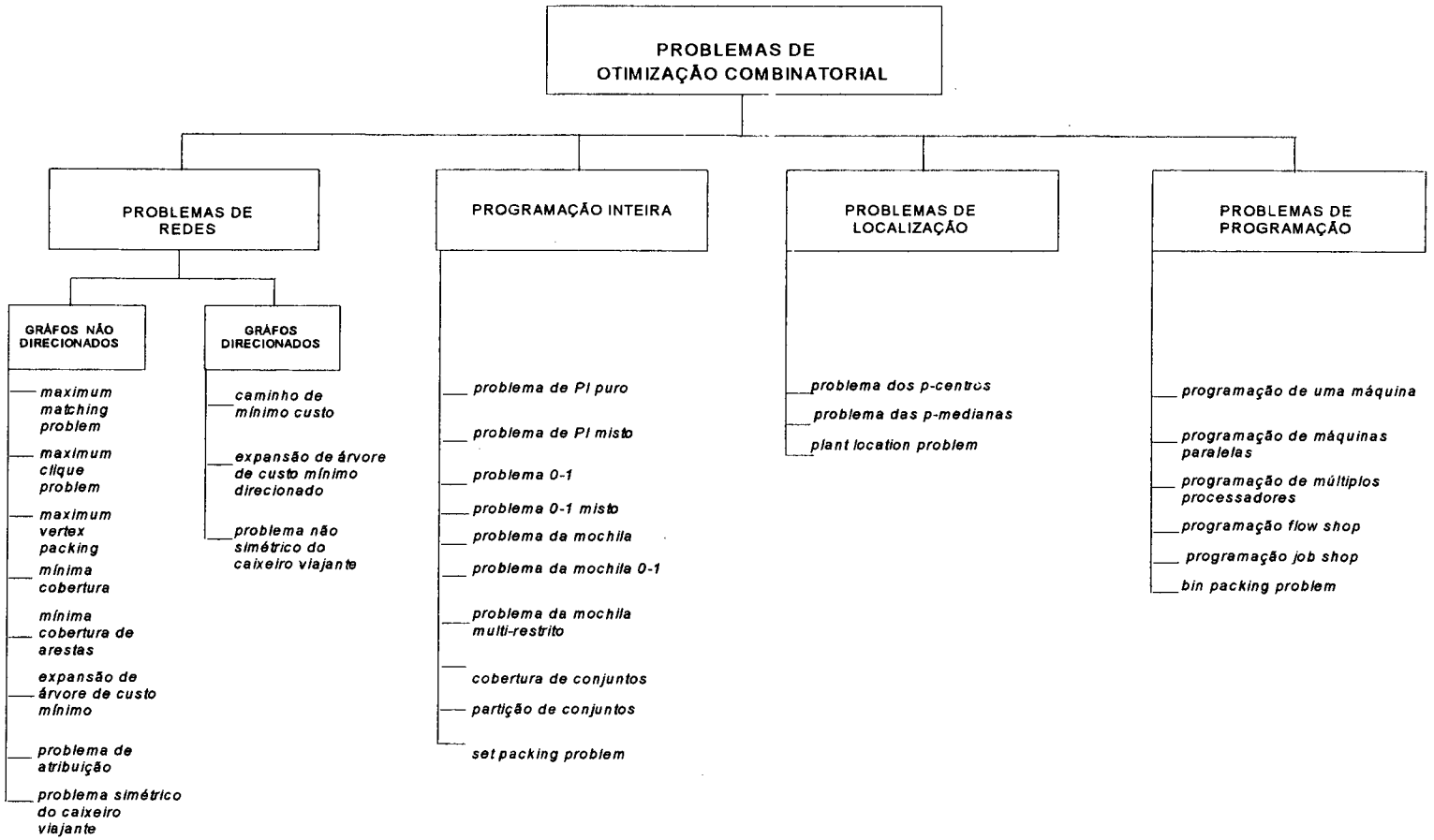


FIGURA 2. 1 CLASSIFICAÇÃO DOS PROBLEMAS DE OTIMIZAÇÃO COMBINATORIAL (ADAPTADO DE IBARAKI, 1988)

Na primeira classe de problemas combinatoriais, Müller-Merbach lista o problema de atribuição linear, o problema dos transportes e o problema de atribuição quadrática.

Exemplos da segunda classe de problemas são: problema do caminho de mínimo custo, *currency arbitrage problem*, o problema do caixeiro viajante e o problema do carteiro chinês.

Nos problemas de seleção, na terceira classe de problemas combinatoriais, o número de exemplos é grande. Alguns deles são: problema de expansão de árvore, cobertura de aresta, cobertura de nó, cobertura de conjunto, problema da mochila, *edge matching problem*, *node packing problem*, *set packing problem*, partição de conjunto, entre outros.

De acordo com Müller-Merbach, alguns tipos padrões da maioria dos problemas combinatoriais do mundo real, consistem de atribuição, seleção e sequenciamento de componentes, recebendo nomes próprios. Entre os quais tem-se: programação de alocação de veículos, problemas do tipo job shop e flow shop, o problema de balanceamento de linha de montagem, alocação de horários e problemas de alocação de equipes. A Figura 2.2 mostra a classificação baseada em Müller-Merbach.

2.4 Um Modelo Geral De Programação

A seguir, baseado em Coffman (1976), é brevemente descrito um modelo geral para os problemas de programação, considerando os recursos, sistemas de tarefas, restrições de sequenciamento e medidas de desempenho.

Na maioria dos modelos de programação, os recursos consistem de um conjunto $P = \{P_1,...,P_m\}$ de processadores. Há também, um conjunto adicional de tipos de recursos $R = \{R_1,...,R_s\}$ solicitados durante a execução de uma tarefa em algum processador. O total dos tipos de recursos R_j é dado po um número positivo m_j. Um sistema geral de tarefas para um dado conjunto pode ser definido como o sistema $(T, \prec, [\tau_{ij}], \{R_j\}, \{w_j\})$ onde:

1. $T = \{T_1,...,T_n\}$ é o conjunto de tarefas a serem executadas,
2. $\prec$ é uma (irreflexiva) ordem parcial definida sobre T, que especifica restrições de precedência operacionais, isto é, $T_i \prec T_j$ significa que T_i deve ser completada antes de que T_j possa começar,
3. $[\tau_{ij}]$ é uma matriz m x n de tempos de execução, onde $\tau_{ij} > 0$ é o tempo requerido para executar T_j, $1 \leq j \leq n$, no processador P_i, $1 \leq i \leq m$,
4. $R_j = [R_1(T_1),..., R_s(T_j)]$, $1 \leq j \leq n$, especifica para o *i-éssimo* componente, o montante de recursos do tipo R_i requeridos ao longo da execução de T_j.
5. w_j , $1 \leq i \leq n$, é interpretado como a taxa de custo. Isto é, o custo de terminar T_i no tempo t é $w_i t$.

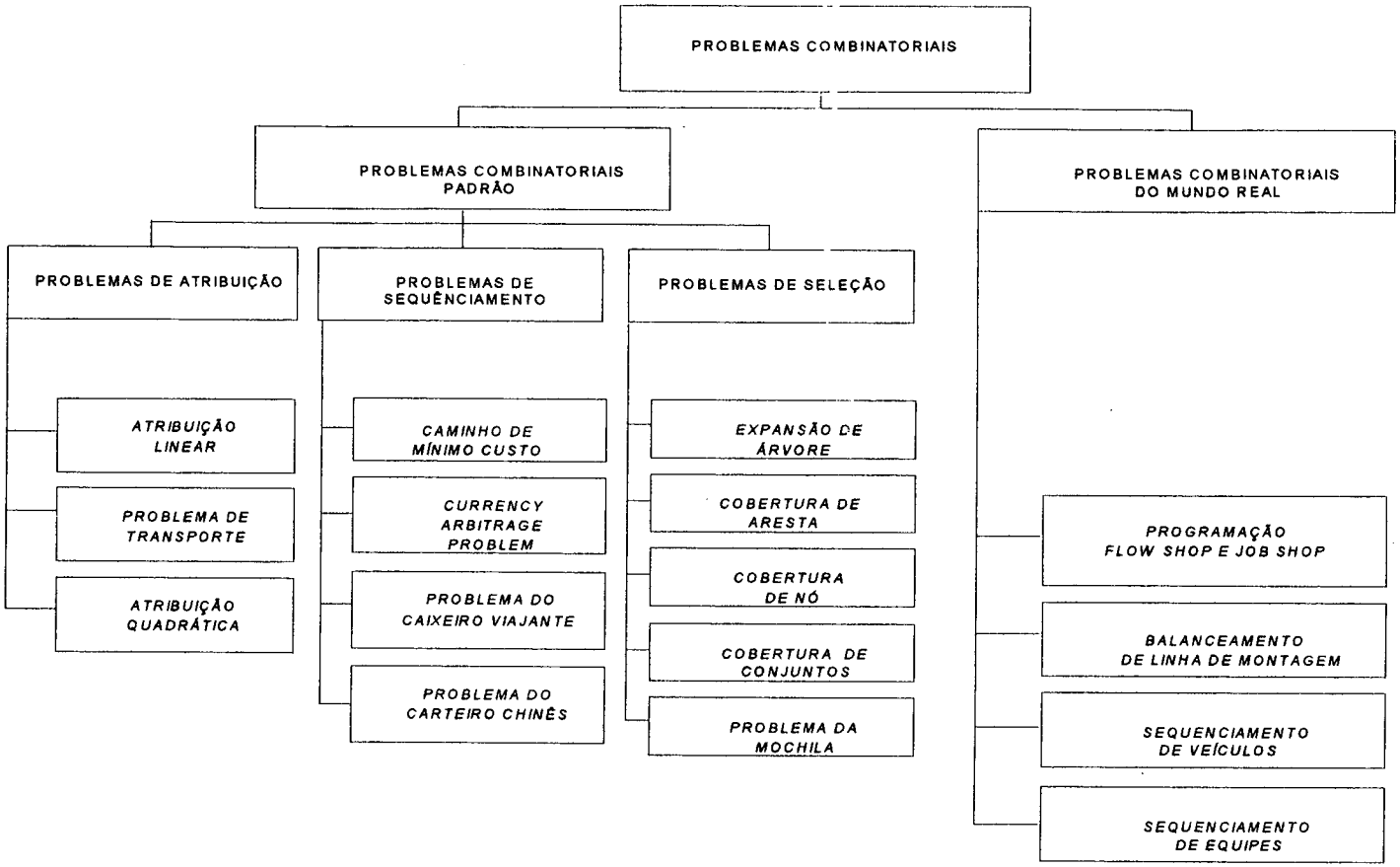


FIGURA 2 2 CLASSIFICAÇÃO DOS PROBLEMAS DE OTIMIZAÇÃO COMBINATORIAL (MÜLLER -MERBACH, 1981)

Por restrições de sequenciamento entende-se uma restrição do algoritmo de programação à uma determinada classe, por exemplo, programação com ou sem interrupção na execução das tarefas. Por outro lado, considera duas principais medidas de desempenho da programação, quais sejam, *comprimento da programação ou tempo máximo de término* e *o tempo médio ponderado de término.*

De acordo com Coffman, os problemas estudados na literatura (de programação da produção) podem ser representados como casos especiais deste modelo.

2.5 Complexidade Dos Problemas De Programação

Dois principais conceitos vem a tona na discussão da complexidade de um problema de otimização, quando é considerada a classe ao qual ele pertence, a saber *classe P* e *classe NP.*

P identifica a classe de problemas para os quais um bom ou eficiente algoritmo, polinomialmente limitado existe, a onde todos os problemas em *NP* podem ser resolvidos usando uma busca recursiva de profundidade polinomial (Lenstra, 1977).

A seguir, algumas idéias e definições são introduzidas para um melhor entendimento dos conceitos de classe de problemas *P* e *NP.*

No campo da teoria da programação, é muito comum encontrar vários problemas que podem ser facilmente resolvidos, enquanto que outros, parecem ser muito difíceis. Antes de discutir se um problema combinatorial é fácil ou difícil, deve-se primeiro definir a fronteira destes dois conceitos.

De acordo com Ibaraki (1988), comumente um problema é dito ser fácil se existe um algoritmo com complexidade de tempo[1] $O(N^k)$ para uma constante k, onde N é o tamanho do problema. Tal complexidade é dita ser de *ordem polinomial*, e o algoritmo é denominado *algoritmo de tempo polinomial.* Por outro lado, se qualquer algoritmo requer uma complexidade não limitada superiormente em N, ele é considerado difícil ou intratável. Por exemplo, ordens típicas não polinomiais são $O(N^{\log N})$ e $O(k^N)$.

Como ponto de partida para a discussão desses problemas e suas dificuldades, primeiramente deve-se definir o que, formalmente, é denominado de um problema de decisão dentro da teoria de otimização combinatorial.

Ullman (1976), define um *problema de decisão* como uma questão que tem uma resposta *sim* ou *não* para uma dada instância do problema. *Instância* é a seleção de valores para os parâmetros de um problema, isto é, são variáveis livres da questão (Ibaraki, 1987).

[1] *Complexidade do tempo* é o número *n* de passos computacionais, necessários para resolver uma dada instância do problema (Ibaraki, 1987).

A complexidade das instâncias de um problema devem ser medidas em relação ao tamanho delas, isto porque a comparação direta dos resultados computacionais para instâncias grandes ou pequenas não fornecerá nenhuma informação importante. O tamanho da instância de um problema é geralmente definido pelo comprimento dos dados de entrada necessários a sua especificação. Por exemplo, para um grafo $G = (V, E)$ com n vértices e m arestas, o comprimento dos dados de entrada de G é $O(n + m)$. Aqui $O(f(n,m))$ lê-se *ordem* $f(n,m)$ e diferentes esquemas de codificação podem ter diferentes comprimentos de entrada, isto é, se o grafo anterior G é representado por uma matriz $n \times m$, o comprimento da entrada requerida é $O(n^2)$ ao invés de $O(n+ m)$ (para maiores detalhes ver Ibaraki,1987).

Agora os conceitos de problemas *P* e *NP* podem ser melhor entendidos após a apresentação destas idéias introdutórias.

Ullman (1976) define a classe ***NP*** (abreviação de nondeterministic polynomial-time) como o conjunto de todos os problemas que podem ser resolvidos por um computador não determinístico[2] num tempo polinomial. Define *P* como a classe de problemas resolvidos por um computador determinístico num tempo polinomial. Em outras palavras, um problema é dito pertencer à classe *NP* se uma instância de tamanho *N* tem um caminho de cálculo cujo comprimento[3] é limitado superiormente por um polinômio em *N*, que responde *sim* se, e somente se, a instância tem resposta *sim* (Ibaraki, 1987).

Como *NP* é uma classe de problemas combinatoriais muito ampla, aquele problema de complexidade mais alta, poderia ser realmente difícil de resolver. Neste contexto, os conceitos de *completação NP* e *redutibilidade* são chaves importantes na identificação destes problemas.

De acordo com Lenstra (1977), um problema *P'* é *redutível* para um problema *P*, escrito $P' \propto P$, se para qualquer instância de *P'* uma instância de *P* pode ser construída num tempo polinomialmente limitado, tal que resolvendo a instância de *P* resolve também a instância de *P'*.

Por outro lado, um problema *P* é denominado de *NP-pesado*, se qualquer *P'* na classe *NP* é redutível para *P*, isto é, não é mais fácil que qualquer problema em *NP*. Então, se um problema *NP-pesado* pertence a *NP*, então *P* é denominado *NP-completo* (Ibaraki, 1977) ou mais formalmente definido como: *P é NP-completo se* $P \in NP$ *e* $P' \propto P$ *para cada* $P' \in NP$ (Lenstra, 1977).

Como conclusão pode ser dito que, reconhecidamente, um problema NP-completo é o mais difícil de NP e, portanto, se um problema NP-completo tem uma solução em tempo

---

[2] *Computação não determinística* é um modelo computacional que executa uma instrução por vez com uma operação fictícia. ESCOLHER (L_1, L_2,...,L_k). Toda vez que o computador encontra esta operação, ele pula para a localização de rótulos L_1, L_2,...,L_k e então executa simultanemanete a operação correspondente. A computação convencional é algumas vezes denominada de *computação determinística* (Ibaraki, 1988).

[3] O tempo usado por um computador não determinístico é definido como o comprimento da sequência mais longa dos estados de transição realizados (Ullman, 1976)

polinomial determinístico, então todos os problemas em NP também tem (Ullman, 1976). A Tabela 2.1, baseada em Lenstra, apresenta alguns problemas NP-completos.

Problema	Observação
3-Satisfiability	Com ao menos três literais por cláusula
Clique	Dado um grafo indireto **G**=(V,E) e um inteiro, tem **G** um sub-grafo completo sobre k vértices?
Ordenação Linear	Dado um grafo indireto **G**=(V,E) e um inteiro k, existe uma função uma-para-uma π ?
Feedback Arc Set	Dado um grafo direto **G**=(V,A) e um inteiro k, tem **G** um conjunto de arcos de retorno de cardinalidade k?
Circuito Hamiltoniano Direto	Dado um grafo direto **G**=(V,A), tem **G** um circuito hamiltonianot?
Caminho Hamiltoniano Direto	Dado um grafo direto **G'**=(V',A'), tem **G** um caminho hamiltoniano?
Circuito Hamiltoniano Indireto	Dado um grafo indireto **G**=(V,E), tem **G** um circuito hamiltoniano?
Problema da mochila	Dados os inteiros positivos $a_1,...,a_t$, existe um sub-conjunto **S**⊂**T** tal que $\Sigma_{i\in S}\, a_i = b$? ($i$∈**S**)
Partição	Dados os inteiros positivos a_1, ..., a_t, existe um sub-conjunto **S**⊂**T** tal que $\Sigma_{i\in S}\, a_i = \Sigma_{i\in T-S}\, a_i$?
3-Partição	Dados os inteiros positivos $a_1,...,a_{3t}$, b, existe uma partição ($T_1,...,T_t$) def **T** tal que $\mid T_j \mid$=3 e $\Sigma_{i\in Tj}\, a_i = b$ para j=1,..,t ?

TABELA 2.1 ALGUNS PROBLEMAS NP-COMPLETOS (ADAPTADO DE LENSTRA 1977)

Os problemas de programação são reconhecidamente caracterizados por uma inerente dificuldade de solução, muitos dos quais são NP-pesados, indicativo da intratabilidade de grandes e, frequentemente, de problemas de tamanho moderados. Neste casos, é provável que seja necessário o emprego de heurísticas para encontrar uma aproximação da solução ótima. Provas sobre a NP-completação de problemas de programação podem ser encontradas em Ullman (1976), Ibaraki (1977) e Kolen e Kroon (1991), entre outros.

2.6 Abordagens Para O Problema De Programação

Muitas vezes, não é simples encontrar uma classificação exata para os problemas de programação, não somente porque existem diferentes versões para um dado problema, mas, porque vários procedimentos para uma questão particular, são caracterizados por premissas diferentes e limitações de aplicação dos modelos desenvolvidos.

A grande maioria de procedimentos de programação, podem ser categorizados em dois principais grupos, baseados nas abordagens empregadas. o primeiro, e de longe a maior categoria, denominado de *procedimentos heurísticos* objetiva produzir programações viáveis *boas.* O segundo grupo consiste dos procedimentos que procuram produzir a programação melhor ou ótima. Estes são denominados de *procedimentos ótimos,* também chamados *exatos* ou *analíticos* porque geralmente envolvem alguma forma de programação matemática ou outro procedimento analítico mais rigoroso.

Dentro de cada um destes grupos há possíveis esquemas de sub-categorização. Os procedimentos heurísticos por exemplo, podem ser divididos em duas classes (Storer, Wu e Vaccari, 1992):

1. *heurísticas específicas,* desenvolvidas para cada tipo de problema, baseadas nas especificações da questão tratada (ex.[4] regras de *dispatching* e sequênciamento e abordagens de Sistemas Especialistas) e,
2. *heurísticas de busca local,* que são formas alternativas de busca do espaço de soluções baseadas no princípio da vizinhança[5], como por exemplo: *hill climbing, steepest descent, Simulated Annealing (SA),* Algoritmos Genéticos (AG) e *Tabu Search.*

Em relação aos procedimentos ótimos, outras duas classes principais podem ser identificadas (MacCarthy and Liu, 1993):

1. *métodos ótimos eficientes:* são métodos que geram programações ótimas em tempo polinomial, e
2. *métodos ótimos enumerativos:* são métodos que envolvem uma enumeração parcial do conjunto de todas as programações possíveis. Esta abordagem é desenvolvida baseada em dois conceitos principais (Day and Hottenstein, 1970): a) o uso de uma técnica de enumeração controlada para considerar todas as soluções potenciais e, b) a eliminação de potenciais soluções inaceitáveis. As mais gerais são formulações de programação matemática, métodos de branch and bound e métodos de eliminação.

2.7 Classificação Dos Problemas de Programação

De acordo com Bellman *et all.* (1982), os problemas de programação podem ser agrupados em três classes básicas:

1. problemas de sequenciamento ou problemas de programação da produção,
2. problema de programação de projetos e,
3. problema de linha de montagem

A seguir cada um dos problemas são brevemente descritos.

---

[4] *Regras de dispatching e sequenciamento* fornecem uma solução para o problema de programação empregando regras locais para a seleção de tarefas (Matsuura *et al.*, 1993, Chryssolouris *et al.*, 1992), enquanto que nos *sistemas especialistas,* as regras aplicadas pelo homem para atribuir as tarefas aos recursos, são codificadas numa base de regras a qual é invocada toda vez que uma decisão de atribuição necessita ser feita (Chryssolouris *et al.*, 1992).

[5] Cada iteração começa de um *estado de solução* (que representa o *conjunto de soluções*). É daqui que as candidatas devem ser determinadas e avaliadas de forma que o estado de solução da próxima iteração (ou próximas iterações) possa ser escolhido. O *conjunto de candidatas* é denominado a *vizinhança* do estado (Müller-Merbach, 1981).

2.7.1 Problema De Programação Da Produção

Tipicamente, o problema de programação da produção envolve um conjunto de produtos que devem ser completados, onde cada produto utiliza um conjunto de operações para ser executado. As operações necessitam de máquinas e recursos materiais e devem ser realizadas seguindo alguma sequência tecnológica. Desenvolver uma programação da produção envolve a seleção de uma sequência de operações (ou rotas de processo) que resultará na realização do produto. Envolve também, a determinação dos recursos necessários para executar cada operação da rota, bem como a determinação das datas em que cada operação da rota começará e terminará (Rodammer e White, 1988). Portanto, a programação da produção é um problema que decide a ordem de execução de todos os produtos em cada máquina e que determina a data de início de cada operação de forma a otimizar uma função objetivo (Bellman *et al.* 1982).

O problema geral de programação da produção pode ser definida como (MacCarthy e Liu, 1993):

Um conjunto de n produtos $\{J_1, J_2, ..., J_n\}$ deve ser processado em m máquinas $\{M_1, M_2, ..., M_m\}$ disponíveis. Um sub-conjunto destas máquinas é requerido para completar o processamento de cada produto. O padrão de fluxo nas máquinas pode ou não ser fixado para algum ou todos os produtos. O processamento do produto J_i na máquina M_i é denominado de uma operação, denotado por O_{ij}. Para cada operação O_{ij}, existe um tempo de processamento t_{ij}. associado. O problema consiste em encontrar uma programação que otimize alguma medida de desempenho.

Está bem estabelecido que a maioria do problemas de programação da produção pertencem a classe dos NP-completos. Trabalhos relativos à complexidade e provas da NP-dureza de novos e velhos e já sedimentados modelos de programação da produção podem ser encontrados em Ullman (1975), Rinnooy Kan (1976); Ullman (1976), Garey *et al.* (1976), Lenstra *et al.* (1977), Lenstra (1977), Ibaraki (1977), Lenstra e Rinnooy Kan (1978), King e Spachi (1980), Goyal e Sriskandarajah (1988), Monma e Potts (1989), Kamoun e Sriskandarajah (1993), Chen e Bulfin (1993). A Tabela 2.2 resume alguns dos problemas de programação de máquinas NP-completos (Ibaraki, 1977; Lenstra, 1977; MacCarthy e Liu, 1993; Kamoun and Sriskandarajah, 1993).

Classificação Dos Problemas De Programação Da Produção

A intenção desta seção não é dar um levantamento exaustivo da literatura de programação da produção, que pode ser encontrado em vários trabalhos de revisão, tais como, Elmaghraby (1968), Bakshi e Arora (1969), Day e Hottenstein (1970), Lenstra *et al.* (1977), Panwalker e Iskander (1977), Godin (1978), Graham *et al.* (1979), King e Spachis (1980), Graves (1981), Frost (1984), Rodammer e White Jr. (1988), Buxey (1989), Hendry e

Kingsman (1989), Baker e Scudder (1990) e MacCarthy B.L. e Liu J. (1993). O objetivo aqui é delinear uma classificação ampla que permita estabelecer o sentido, direção e perspectiva de pesquisas conduzidas na área, assim como oferecer um resumo das principais abordagens e paradigmas desenvolvidas.

Função Objetivo	Problema	Observações
C_{max}	*F, m=2*	
	F, m=2, $r_j \geq 0$	com diferentes ready times
	F, m=3	
	F, tree, m=2	
	F, no wait	m máquinas
	J, m=2, $n_i \leq 3$	cada produto pode ter ao menos 3 operações
	J, m=3, $n_i \leq 2$	cada produto pode ter ao menos 2 operações
	I, m=2	
	I, m=2, prec, $p_i \leq 2$	
	I, prec, $p_i = 1$	m máquinas
$\Sigma w_i C_i$	*//, prec, $p_i = 1$*	
	//, prec, $w_i = 1$	
	//, $r_i \geq 0$, $w_i = 1$	
	//, $C_i \leq d_i$	Produtos atrasados não são permitidos
	F, m=2, $w_i = 1$	
	F, no wait, $w_i = 1$	m máquinas
	I, m=2	
	I, m=2, prec, $p_i \leq 2$, $w_i = 1$	
	I, prec, $p_i = 1$, $w_i = 1$	m máquinas
L_{max}	*//, $r_i \geq 0$*	com diferentes *ready times*
	F, m=2	
	I. m=2	
$\Sigma w_i T_i$	*//*	
	//, prec, $p_i = 1$, $w_i = 1$	
	//, $r_i \geq 0$, $w_i = 1$	
	F, m=2, $w_i = 1$	
	I, m=2, $w_i = 1$	
$\Sigma w_j N_T$	*//*	
	//, prec, $p_i = 1$, $w_i = 1$	
	//, $r_i \geq 0$, $w_i = 1$	

TABELA 2.2 PROBLEMAS DE PROGRAMAÇÃO NP-COMPLETOS
(ADAPTADO DE IBARAKI 1977, LENSTRA 1977, MACCARTHY AND LIU 1993, KAMOUN AND SRISKANDARAJAH 1993)

Notação

C_{max}	Tempo máximo de processamento (makespan)
$S\ w_j C_j$	Soma ponderada dos tempos de processamento
L_{max}	Máximo atraso
$S\ w_j T_j$	Soma ponderada de atraso
$S\ w_j N_T$	Soma ponderada de produtos atrasados
n	Número de produtos
m	Número de máquinas
prec	Restrição de precedência geral
tree	Restrição de precedência do tipo árvore
no wait	Tempo de não espera para produtos entre seus tempos de início e término
//	Problema de uma máquina
F	Problema de programação flow shop
J	Problema de programação job shop
I	Porblema de programação de máquinas paralelas
r_i	Tempo de início
p_i	Tempo de processamento
n_i	Limite superior, constante sobre o número de operações por produto
w_i	Peso
$C_i \leq d_i$	Todos os produtos devem realizar suas data limites ("deadlines")

Diversos esquemas tem sido propostos para a categorização dos problemas de programação da produção. Conway *et al.* 1967 (in: MacCarthy B.L. e Liu J. 1993) sugere um esquema de classificação baseado em quatro descritores *A/B/C/D*, onde *A* representa o número de produtos, *B* representa o número de máquinas, *C* representa o padrão de fluxo e futuras restrições tecnológicas e de administração e, *D* representa o critério a ser otimizado Por exemplo, *n/m/J/Cmax* refere-se ao problema job shop com *n* produtos e *m* máquinas que objetiva minimizar o tempo total de processamento (makespan).

Lenstra (1977) introduz um parâmetro λ ao terceiro elemento desse esquema tradicional, para representar algumas variações relevantes. A notação *n/m/*λ*/k*, é utlizada para representar nos dois primeiros parâmetros o número de produtos e de máquinas respectivamente, *l* assume um valor de acordo com o padrão de fluxo e o quarto elemento indica o critério de otimalidade.

Lenstra e Rinnooy Kan (1978) caracteriza cada problema de programação em outra classificação de quatro parâmetros $\alpha/\beta/\gamma/\delta$, onde α indica o número de máquinas, β indica ou o tipo de restrições de precedências (*prec* ou *tree*) ou a ausência delas, γ indica ou a restrição que o *tempo de processamento* $p_i \in \{\gamma\}$ para todo $l \in l$, ou a ausência da mesma, δ representa o critério de otimalidade.

Graham *et al.* (1979) apresenta uma notação baseada em três parâmetros $\alpha/\beta/\gamma$ onde o primeiro elemento define o número de máquinas e o padrão de fluxo, β indica alguma restrição sobre os produtos e γ define o critério de programação.

O esquema aqui proposto para a classificação dos problemas de programação é baseado nas características gerais dos problemas e as hipóteses usuais feitas sobre os produtos, máquinas e tempos de processamento. Esta categorização pode ser considerada como uma macro-divisão dos problemas de programação da produção.

De acordo com King e Spachis (1980), Bellman *et al.* (1982) e Forst (1984) as premissas básicas para os produtos, máquinas e tempos de processamento geralmente assumidas na literatura são:

1. *Premissas para os produtos*

   - Todos os *n* produtos a serem realizados estão simultaneamente disponíveis na data zero.
   - As operações não podem ser interrompidas. Qualquer operação, uma vez começada deve continuar até a sua completa realização.
   - Cada produto consiste de *m* operações que devem ser processadas em *m* máquinas.

- O processamento de qualquer operação não pode ser feito por mais do que uma máquina por vez.
- Não há ordem de prioridade entre os produtos e cada um tem o mesmo grau de importância
- É permitido estoques ilimitados de produtos em processo ou tempos de espera intermediários
- Uma data final (due-date) pode ser atribuídas a cada produto.
- Cada produto pode ser processado em cada máquina uma única vez

2. *Premissas para as máquinas*

- A capacidade das máquinas e da mão-de-obra é fixa
- Todas as *m* máquinas estão disponíveis no início. Quebras e reparações não são permitidas durante o processo.
- Qualquer máquina pode processar qualquer produto.
- Existe uma máquina de cada tipo ou são arranjadas para trabalhar em paralelo uma independentemente da outra.
- Uma máquina não pode executar mais do que uma operação por vez
- Não é possível o intercâmbio de máquinas (flexibilidade).

3. *Premissas para os tempos de processamento*

- O tempo de processamento de cada operação é dado no início e permanece constante, independentemente da ordem de execução.
- O tempo de preparação e de transporte é incluído no tempo de processamento de uma operação.

Vários problemas de programação da produção são caracterizados relaxando ou restringindo algumas destas premissas. Uma grande gama de resultados e variações existem para um vasto conjunto de especificações ou restrições dos problemas. Por exemplo, *restrições de precedências comuns* entre produtos (J_i precede J_j), *restrições de precedências* entre produtos do tipo *árvore* (o grafo de precedência pode ser como ramificações), possíveis *datas de liberação* não iguais para os produtos, não tem *datas de espera* (*no wait*) para os produtos entre o seu início e o tempo de realização, um limite superior constante para o *número de operações* por produtos, um limite superior constante para o *tempo de processamento*, rotas de *processo estáticas,* onde cada produto é executado em cada máquina exatamente um vez em estrita sequência tecnológica (entre produtos) ou, *rotas de processamento dinâmicas,* onde a rota de processamento pode mudar

dinamicamente pelo estado da planta (ex. disponibilidade de recursos) ou pelas condições do trabalho (ex. necessidade de retrabalho).

Consequentemente, com isto em mente, um esquema para classificar os problemas de produção é ilustrado na Figura 2.3. Este esquema mostra que os problemas de programação podem ser classificados de acordo com:

1. As restrições tecnológicas dos produtos (Coffman Jr. 1976)

- *O caso de processamento sem interrupção:* cada operação uma vez iniciada deve ser executada até o seu término;
- *O caso com interrupção:* uma tarefa pode ser interrompida e removida do processador sob a hipótese de que, eventualmente, todo o tempo de processamento será realizado, e que não há perda do tempo já empregado devido à interrupção, ou seja, as tarefas interrompidas reiniciam a suas execuções no ponto que foram interrompidas.

2. O critério de programação (Graves 1970, Rodammer e White 1988 e MacCarthy e Liu, 1993)

O critério indica a medida sobre a qual as programações, sob qualquer conjunto de premissas, são avaliadas. Neste sentido, duas classes de problemas podem ser identificadas:

- *Problemas de único critério:* as formulações tradicionais de programação da produção, geralmente especificam um único critério, usualmente associados com custos ou desempenhos das programações. Critérios associados aos custos incluem custos de permanência de estoques, produção variável e custo de atrasos etc. Medidas comuns do desempenho da programação estão relacionados com a utilização eficiente dos níveis dos recursos de produção (makespan, *Cmax*), rápida resposta à demanda (tempo médio de processamento C, tempo médio de fluxo F, tempo médio de espera W) e cumprimento de datas limites (atraso médio T, atraso máximo T_{max}, número de produtos atrasados N_T).
- *Múltiplos critérios:* ambientes atuais de produção abrangem múltiplas, conflitantes e algumas vezes não comensuráveis restrições e objetivos de desempenho, como minimizar stresses operacionais, melhorar a estabilidade da programação, redução da ociosidade de máquinas para diminuir filas e estoques ou para assegurar a confiabilidade.

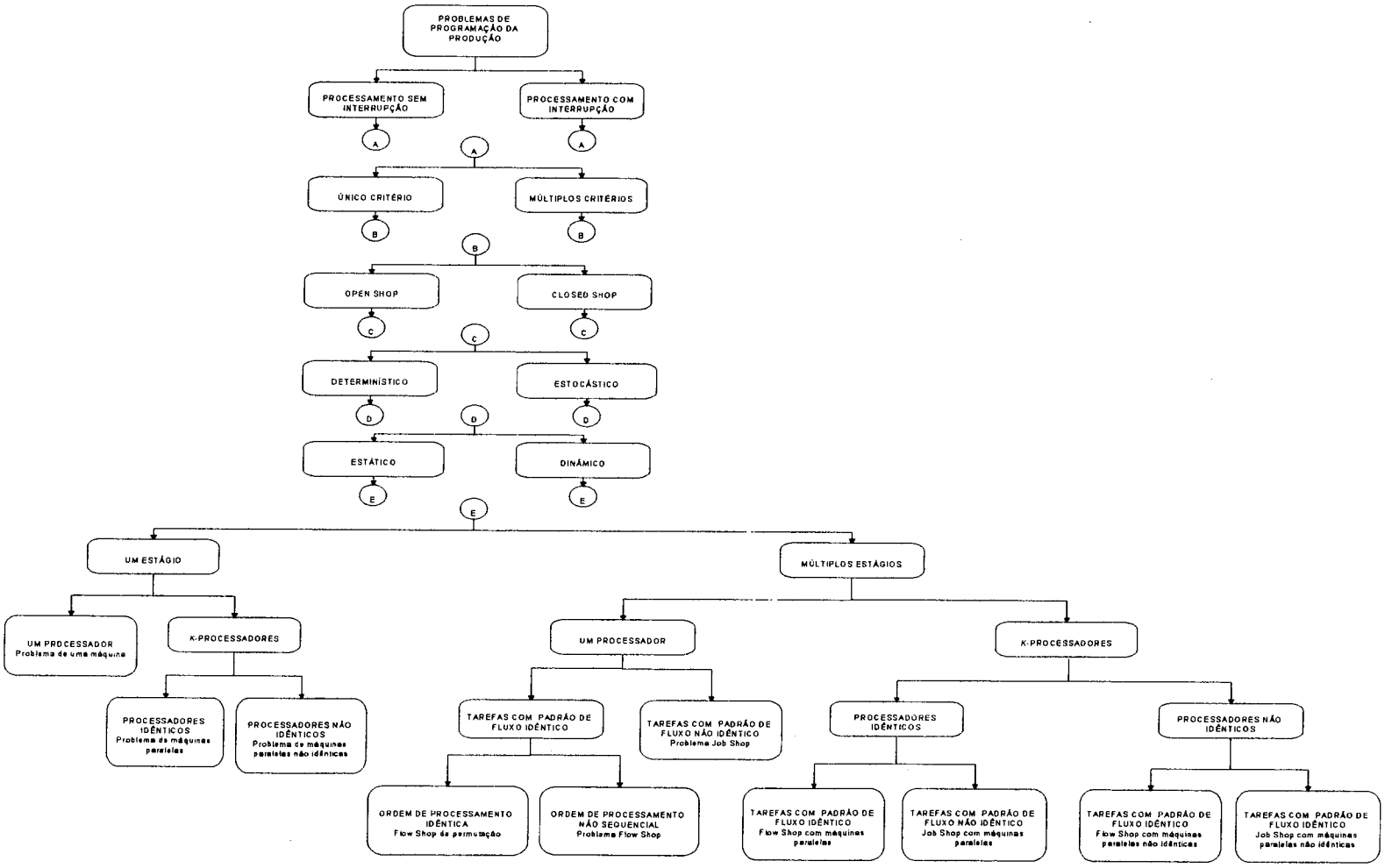


FIGURA 2.3 REPRESENTAÇÃO ESQUEMÁTICA PARA PROBLEMAS DE PROGRAMAÇÃO DA PRODUÇÃO

3. A geração de solicitações (Graves 1970)

- *open shop:* todas as ordens de produção são solicitações dos clientes e não há estoques finais
- *closed shop*: todas as solicitações dos clientes são atendidas pelos estoques e as tarefas de produção são geralmente, resultados de decisões de reposição de estoques. Este problema é mais conhecido como *problema de tamanho de lote.*

Outros nomes empregados para esta divisão são: sistemas *Make-to-Order* e *Make-to-Stock* respectivamente (Hendry e Kingsman, 1989).

4. O ambiente de programação (Rodammer e White 1988; Graves 1970; Bellman *et al.* 1982)

- *Determinístico:* o problema de programação é definido com respeito a um conjunto finito de solicitações totalmente especificadas, isto é, tempos de processamentos, tempos de liberação e datas de entrega são determinísticos. Produtos, máquinas, mão-de-obra e recursos materiais estão disponíveis para todos ao longo do processo.
- *Estocásticos:* o problema estocástico é definido levando em consideração eventos e disturbios aleatórios tais como: quebras de máquinas, indisponibilidade de operador e de material, tempos variáveis de processamento, de liberação e de entrega.

5. A natureza da chegada do produto (Day e Hottenstein 1970)

- *Caso estático:* todos os produtos são simultaneamente iniciados.
- *Caso dinâmico:* os produtos estão constantemente entrando e saindo da fila, de acordo com algum processo estocástico

6. A complexidade de processamento (Graves 1970, MacCarthy e Liu 1993)

A complexidade de processamento está relacionada com o número de passos associados com cada tarefa de produção

- *Um estágio com um processador:* todas as tarefas requerem um passo de processamento que deve ser executado em um elemento de produção. este problema é comumente denominado de *problema de uma máquina.*
- *Um estágio com processadores paralelos:* as tarefas podem ser executadas em *k* processadores idênticos[1] ou não idênticos num único estágio. Cada produto necessita uma, e somente uma destas máquinas

[1] *Máquina idêntica* é um equipamento alternativo num dado estágio que permite executar certa operação com tempo de processamento idêntico e, ainda, características e demanda de recursos iguais.

- *Múltiplos estágios, flow shop:* para um problema de múltiplos estágios, cada tarefa requer ser processada num conjunto de máquinas distintas. Nos problemas flow shop, cada produto tem um padrão de fluxo idêntico. Um problema flow shop no qual a ordem de processamento dos produtos em todas as máquinas é restrito de forma a ser a mesma é denominado de *flow shop de permutação.*

- *Múltiplos estágios, job shop:* aqui cada produto tem o seu próprio padrão de fluxo ou rota específica através das máquinas. Isto quer dizer que não há restrição no passos de processamento para as tarefas, e que rotas alternativas para os produtos são permitidas.

- *Múltiplos estágios, flow shop com máquinas paralelas:* é um problema flow shop onde há k_i máquinas idênticas ou não idênticas, em um ou em todos os estágios ($i = 1,2,...,m$), e qualquer produto requerendo este estágio deve ser processado em uma, e somente uma, destas máquinas.

- *múltiplos estágios, job shop com máquinas paralelas:* é um problema job shop onde há k_i máquinas idênticas ou não idênticas, em um ou em todos os estágios ($i = 1,2,...,m$), e qualquer produto requerendo este estágio deve ser processado em uma, e somente uma, destas máquinas. A situação mais geral é o problema job shop em máquinas paralelas não idênticas.

Como resumo do esquema proposto para a categorização dos problemas de programação da produção, alguns trabalhos são listados nas tabelas 2.3 à 2.6. Deve ser observado que a presença de restrições adicionais foram omitidas dada a grande variedade de problemas. As referências citadas tem a intenção de ser exemplos e não são necessariamente exaustivas.

Principais Objetivos Usados Para Medir O Desempenho Da Programação	
C_j	*Tempo de processamento*
$\overline{C}$	*Tempo médio de processamento*
$S\ w_j C_j$	*Soma ponderada do tempo de processamento*
C_{max}	*Tempo máximo de processamento (makespan)*
F_j	*Tempo de fluxo*
$S\ w_j F_j$	*Soma ponderada do tempo de fluxo*
$\overline{F}$	*Tempo de fluxo médio*
$Sw_j\ \overline{F}$	*Soma ponderada do tempo médio de fluxo*
W_j	*Tempo de espera*
$\overline{W}$	*Tempo médio de espera*
$S\ w_j\ W_j$	*Some ponderada do tempo de espera*
T_j	*Tardeza*
$\overline{T}$	*Tardeza média*
T_{max}	*Tardeza máxima*
$S\ w_j T_j$	*Soma ponderada de tardeza*
L_j	*Atraso*
$\overline{L}$	*Atraso médio*
L_{max}	*Atraso máximo*
$S\ w_j\ L_j$	*Soma ponderada de atraso*
N_T	*Número de produtos atrasados*
$S\ w_j\ N_T$	*Soma ponderada de produtos atrasados*
$E_j\ /\ T_j$	*Cedo e Tarde*
St_j	*Tempo de início*
$S\ w_j\ E_j\ /\ w_j\ T_j$	*Soma ponderada de cedos e tardes*
VAR	*Variância do tempo de processamento*
COVAR	*Coeficiente de variação*
BICRET	*Tempo médio de processamento e variância do tempo de processamento*
FV	*Variância do tempo de fluxo*
CTV	*Variância média do tempo de processamento*
NPV	*Valor presente da rede*
MWAD	*Desvio absoluto ponderado do tempo de processamento da data de realização*
f_{max}	*Custo máximo (ou total)*
p_{max}	*Lucro máximot*

Nota: Ver King e Spachis (1980) para equivalências importantes entre medidas de desempenho comumente utilizadas.

TABELA 2.3 PROBLEMAS DE UM ESTÁGIO COM UM PROCESSADOR

Função Objetivo	Método	Referência	Observações‡
C_j	ótimo	Smith, 1956, (veja McCarthey e Liu 1993)	
	ótimo	Glazebrook, 1976 (veja Forst, 1984)	Tempo de processamento estocástico
	ótimo	Szwarc et ai. 1988	
	ótimo e heurística	Chandru et al. 1993	Batch processing machine
$w_j C_j$	ótimo	Held e Karp, 1962 (veja MacCarthey e Liu, 1993)	
$w_j C_j$ e F_j	ótimo	Cheng, 1991	
$\overline{C}$		Rothkopf, 1966 (veja Forst, 1984)	Tempo de processamento estocástico
C_{max}	ótimo	Dauzere-Peres, 1993	
	heurística	Alidaee, 1990	
	ótimo	Glazebrook, 1992	Tempo de processamento estocástico, interrupção e não interrupção
	ótimo	Chu, 1992	datas de liberação desiguais
C_{max} e N_T	ótimo	Kiran e Unal, 1991	Múltiplos critérios
C_{max} e desvio da programação	heurística	Wu et al. 1993	Múltiplos critérios. Permite descontinuidade na máquina.
$\Sigma w_j C_j$	ótimo	Smith, 1956, (veja MacCarthey e Liu, 1993)	
	ótimo	Horn, 1972 (veja King e Spachis, 1980)	
	ótimo	Sidney, 1975; 1977 (veja King e Spachis, 1980)	
	heurística	Morton & Dharan, 1978 (veja MacCarthey e Liu, 1993)	
	heurística	Weiss, 1981 (veja MacCarthy e Liu, 1993)	
CTV	heurística - genetic algorithms	Gupta et al. 1993	
	ótimo	De et al. 1992	
L_{max} ou T_{max}	ótimo	Jackson, 1955 (veja MacCarhy e Liu, 1993)	
L_{max}	ótimo	Lawler, 1973 (veja King e Spachis, 1980)	
	heurística	Lee, 1991	
	ótimo	Leon e Wu, 1992	
L_{max} , T_{max}		Crabill e Maxwell, 1969 (veja Forst, 1984)	Tempo de processamento estocático e "due dates".
T_{max}		Banerjee, 1965 (veja Forst, 1984)	Tempo de processamento estocástico.
	ótimo	Hodgson, 1977 (veja Forst, 1984)	Tempo de processamento estocástico.
	ótimo	Sen e Borah, 1991.	
	ótimo	Potts e Van Wasenhover, 1992	Interrupção e não interrupção nos produtos.
	heurística	Holsenbank e Russel, 1992	
	ótimo	Chu, 1992	Datas de liberação desiguais

TABELA 2.3 CONTINUAÇÃO...

Função Objetivo	Método	Referência	Observações‡
$\overline{T}$	heurística	Wilkinson & Irwin, 1971 (veja MacCarthey e Liu, 1993)	
	heurística	Frey et al. 1971 (veja MacCarthey e Liu, 1993)	
	heurística	Panwalkar et al. 1993	
N_T	ótimo	Moore, 1968 (veja MacCarthey e Liu, 1993)	
	ótimo	Balut, 1973 (veja Forst, 1984)	Tempo de processamento estocástico.
	ótimo	Adiri et al. 1991	Interrupção devido a quebra de máquina.
$w_j N_T$	ótimo	Potts e Wassenhove, 1988	
$w_j T_j$	ótimo	Lawler, 1964 (veja King e Spachis, 1980)	Tempos de chegada diferentes.
	ótimo	Blau, 1973 (veja Forst, 1984)	Tempo de processamento estocástico e "due dates".
	ótimo	Shwimer, 1972 (veja MacCarthey e Liu, 1993)	
	heurística	Chambers et al. 1991	
	ótimo	Szwarc e Liu, 1993	
$\overline{F}$	ótimo	Conway, 1967 (veja MacCarthey e Liu, 1993)	
	heurística	Liu e MacCarthy, 1990	
$w_i \overline{F}$	ótimo	Mason e Anderson, 1991	
FV	heurística	Gupta et al. 1990	
VAR,COVAR,BICRIT	heurística - simulated annealing	Mittenthal et al. 1993	Programação ótima de forma V.
E_j / T_j	ótimo	Hall et al. 1991	
	ótimo	Davis e Kanet, 1993	
	ótimo	Szwarc, 1993	
$E_{j\,j}/T_j$ e ST_j	heurística	Sung e Joo, 1992	Produtos com tempos dinâmicos.
E_j/T_j e custo de entrega de produtos atrasados	ótimo	Herrmann e Lee, 1992	
$w_j E_j / w_j T_j$	ótimo	Hall e Posner, 1991	
T_{max} e SUPj	heurística - Tabu Search	Laguna, et al. 1991, (veja Barnes e Laguna, 1993)	
f_{max}	ótimo	Gordon, 1993	Interrupção.
P_{max}	ótimo	De et al. 1993	Tempo de processamento aleatório e "deadline".
Min: produção esperada e custo de deslocamento	heurística	Leachman e Gascon, 1988	Closed shop - demanda estocástica.
Min: custo médio de preparação, delocamneto e fora de estoque	controle ótimo	Gallego, 1990	Closed shop - demanda aleatória.
Min: custo de produção	ótimo	Ibrahim e Thomas, 1991	Closed shop.
Min: custo total	ótimo	Zaheng at al. 1993	
NPV	heurística	Lawrence, 1991	
MWAD	heurística	Nandkeolyar et al. 1993	Chegada estocástica de produtos

‡ A menos que seja indicado o contrário, assume-se que os problemas sejam determinísticos, não interrompíveis, único objetivo, open shop e caso estático

TABELA 2.4 PROBLEMA UM ESTÁGIO - PROCESSADORES PARALELOS

Função Objetivo	Método	Referência	Observações‡
w_jF_j , $\overline{F}$	Ótimo	McNaughton, 1959 (in: Graves, 1981)	
$w_j F_j$	Ótimo	Eastman et al. 1964 (in: Graves, 1981)	
	Heurística	Baker e Merten, 1973 (in: Graves, 1981)	
	Ótimo E Heurística	Dobson e Karmarkar, 1989	
	Heurística	Webster, 1993	
	Ótimo	Webster, 1992	
	Heurística - Tabu Search	Barnes e Laguna, 1992 (in: Barnes e Laguna, 1993)	
$\overline{F}$	Ótimo	Horn, 1973 (in: Graves, 1981)	Processodores não idênticos.l
$w_j \overline{F}$	Heurística	Herrbach e Leung, 1990	Com interrupção.
$w_j T_j$	Ótimo	Root, 1965; Barnes e Brennan, 1977; Nunnikhoven e Emmons, 1977; (In: Graves, 1981)	
	Ótimo	Elmaghraby e Park, 1974	
	Heurística	Arkin e Roundy, 1991	
	Heurística - Neural Network	Lee e Kim, 1993	
$w_j Ej$	Heurística - Tabu Search	Laguna e Gonzalez-Velarde, 1991 (in: Barnes e Laguna, 1993)	
C_j	Ótimo	Conway et al., 1967; Coffman e Graham, 1972; Baker, 1974; (in: King e Spachis, 1980)	
$w_j C_j$	Ótimo	Lawler, 1964 (in: King e Spachi, 1980)	
C_{max}	Heurística	Barker, 1974, (in: MacCarthy e Liu, 1993)	
	Ótimo	Gonzales e Sahni, 1978 (in: King e Spachis, 1980)	Com interrupção.
	Ótimo	Dobson, et al. 1989	Máquinas heterogêneas.
	Ótimo	Hu, 1961 (in: MacCarthy e Liu, 1993)	
	Heurística	Kulkami e Chimento Jr., 1992	Tempo de processamento estocástico, processodores sujeitos a falhas e reparações (interrupção).
	Ótimo	Van de Vel e Shijie, 1991	Máquinas uniformes operando com diferentes velocidades.
	Heurística	Rajgopal e Bidanda, 1991	
	Heurística	Blocher e Chand, 1991	
C_{max} ou $\overline{C}$	Heurística	Guinet, 1993	
C_{max}	Heurística	França et al. 1994	
C_{max} , F, L_{max}	Ótimo	Su e Sevcik, 1978	Interrupção, caso dinâmico.
C_{max} e L_j e C_{max} do segundo processador de maior tempo de execução	Ótimo	Epstein et al. 1992	Múltiplos objetivos.
L_{max}	Ótimo	Bratley et al. 1975 (in: King e Spachis, 1980)	
N_T	Ótimo e Heurística	Süer e Báez, 1993	
$\overline{T}$	Heurística	Ho e Chang, 1991	
Min: a) custo total de processamento mais F, b) custo total de processamento mais $w_i Ej$ e $w_j T_j$	Ótimo	Alidaee e Ahmadian, 1993	Máquinas não idênticas e tempos de processamento variáveis.
Min: custo de combinação	Ótimo	Dondeti e Emmons, 1992	Processadores não idênticos.
Max: retorno total	Heurística	So, 1990	
programação viável para satisfazer a demanda	Heurística - Simulated Annealing	Jeffcaot e Bulfin, 1993.	Restrição de recursos.
Min: produção média e custo de estoques	Heurística	Carreno, 1990	Closed shop.

‡ A menos que seja indicado o contrário, assume-se que os problemas sejam determinísticos, não interrompíveis, único objetivo, open shop e caso estático

TABELA 2.5 PROBLEMAS MÚLTIPLOS ESTÁGIOS - FLOW-SHOP

Função Objetivo	Método	Referência	Observações‡
C_{max}	Ótimo	Johnson, 1954; Gilmore e Gomory, 1864; Reddi e Ramamoorthy, 1972 (in: King e Spachis, 1980)	
	Ótimo	Wagner, 1959; Kusiak, 1986; Chow, 1989 (in: MacCarthy e Liu, 1993)	
	Ótimo	Smith e Dudek, 1967, Lageweg et al., 1978; Swarc, 1973; Srikar e Ghosh, 1986; Stafford, 1988; (in: MacCarthy e Liu, 1993)	Flow shop de permutação.
	Ótimo	Ignal e Schrage, 1965; Lomnicki, 1965; McMahon e Burton, 1967; Ashour, 1970; Gupta, 1971; Szwarc, 1971 (in: Graves 1981)	
	Ótimo	Makino's, 1965; Talwar, 1967; Cunningham e Dutta, 1973; Mittal e Bagga, 1977; Prasad, 1981 (in: Frost, 1984)	Tempo de processamento estocático.
	Ótimo	Vickson e Alfredson, 1992	Uso de batches de transferência.
	Ótimo	Karabati et al. 1992	Flow shop de permutação.
	Ótimo	Tsujimura et al. 1993	Tempos de processamento incertos.
	Ótimo e heuristics	Lee et al. 1993	Flow shop tipo linha de montagem.
	Ótimo	Wagneur e Sriskandarajah, 1993	Flow shop de permutação.
	Ótimo	Nowicki, 1993	Flow shop de permutação, tempos de processamento controláveis.
	Heurística	Campbell et al. 1970; Dannenbring, 1977 (in: Graves, 1981)	
	Heurística	Palmer, 1965; Ashour, 1970; Gupta, 1971; Bonney e Gundry,1976; (in: MacCarthey e Liu, 1993)	
	Heurística	Pinedo, 1982 (in: Forst, 1984)	Tempos de processamento estocásticos.
	Heurística - Tabu Search	Widmer e Hertz, 1989	Flow shop de permutação.
	Heurística	Proust et al. 1991	
	Heurística	Cao e Bedworth, 1992	
	Heurística - variants of Simulated Annealing e Tabu Search	Werner, 1993.	Flow shop de permutação.
	Heurística - Genetic Algorithm	Vempati et al. 1993	
C_{max}, *Tempo de preparação esperado*	Ótimo	Bagga, 1970 (in: Frost, 1981)	Tempos de processamento estocásticos.
C_{max} *e* F	Ótimo	Rajendran, 1992	Bi-critérios.
$\overline{F}$	Ótimo	Cadambi e Sathe, 1993	
C_j	Ótimo	Karabati e Kouveiis, 1993	Flow shop de permutação.
$\overline{T}$	Ótimo	Kim, 1993	
Achar requerimentos de demanda ao mínimo custot	Ótimo	Zangwili, 1969; Love, 1972 (in: Graves, 1981)	Closed shop.
	Heurística	Lambrecht e Vander Eecken, 1978 (in: Graves, 1981)	Demanda variável.
Min: cutso total esperado		Forst, 1983	Tempos de processamento estocásticos.

‡ A menos que seja indicado o contrário, assume-se que os problemas sejam determinísticos, não interrompíveis, único objetivo, open shop e caso estático

TABELA 2.6 PROBLEMAS MÚLTIPLOS ESTÁGIOS - JOB-SHOP

Objetivo Função	Método	Referência	Observações‡
C_{max}	Ótimo	Jonhson, 1954; Ashour e Hiremath, 1973; Barker, 1985; (in: MacCarthy e Liu, 1993)	
	Ótimo	Jackson, 1956; Szwark, 1960;	
	Ótimo	Giffer e Thompson, 1960; Brooks e White, 1965; Greenberg, 1968; Balas, 1969; Scharage, 1970; Charlton e Death, 1970; Florian, 1971; Ashour et al., 1974; Lageweg et al. 1977 (in: Graves, 1981)	
	Ótimo	Carlier e Pinson, 1989	
	Ótimo	Applegate e Cook, 1990	
	Ótimo	Liao e You, 1992	
	Ótimo	Dauzere-Peres e Lasserre, 1993	Lot streaming.
	Heurística	Ashour, 1967; Ashour e Hiremath, 1973; (in: MacCarthy e Liu, 1993)	
	Heurística	Pinedo, 1981 (in: Forst, 1984)	tempo de processamento estocástico
	Heurística	Adams et al. 1988	
	Heurística - Tabu Search	Taillard, 1989; (in: Barnes e Laguna, 1993)	
	Heurística - Tabu Search	Widmer, 1991	
	Heurística - Simulated Annealing	Matsuo et al. 1988; (in: Dell'Amico e Trubian, 1993)	
	Heurística	Storer at al. 1992	
	Heurística - Simulated Annealing	Van Laarhoven et al. 1992	
	Heurística - Algoritmos Genéticos	Biegel e Davern, 1990	
	Heurística - Algoritmos Genéticos	Nakano e Yamada, 1991	
	Heurística	Dauzere-Peres e Lasserre, 1993	
	Heurística - Tabu Search	Dell'Amico e Trubian, 1993	
	Heurística	Gupta e Tunc, 1991	Máquinas paralelas.
L_{max}	Ótimo	Townsend, 1977 (in: King e Spachis, 1980)	
	Ótimo	Veitnot, 1969 (in: Graves, 1981)	Closed shop, demanda variável.
	Heurística	McLAren, 1976; McLaren e Whybark, 1976; Biggs at al. 1977; Blackburn e Millen, 1979; Graves, 1981 (in: Graves, 1981)	Closed shop, demanda variável.
$w_j T_j$	Ótimo	Fisher, 1973 (in: Graves, 1981)	
T_{max}	Ótimo	Chu et al. 1992	
Achar requerimentos de due date; max. utilização de máquinas	Huerística	Doctor et al. 1993	Linha de montagem job-shop (inclui operações seriais e paralelas).
Min: custo total	Ótimo	Crowston et al. 1973; Schwarz e Schrage, 1975 (in: Graves, 1981)	Closed shop.

‡ A menos que seja indicado o contrário, assume-se que os problemas sejam determinísticos, não interrompíveis, único objetivo, open shop e caso estático

2.7.2 Problema De Programação De Projetos

O problema de programação de projetos tem sido objeto de interesse desde o advento das técnicas PERT e CPM. A questão básica desta área é encontrar uma programação viável tal que algum objetivo de administração seja otimizado. O problema surge quando as quantidades dos recursos solicitados pelas atividades são limitados. Devido a isto, decisões sobre o sequenciamento das atividades são requeridas, resultando em incremento da duração do projeto além daquela determinada sem a consideração deste tipo de limitação. O objetivo é tomar as decisões de sequenciamento, sujeito a restrições de recursos e precedência, de modo a minimizar este incremento da duração. (Bell and Han, 1991). Este tipo de problema é denominado *programação de projetos com restrição de recursos.*

Porém, sob restrição de recursos, obter uma programação ótima ou viável para a maioria dos problemas práticos de programação é NP-pesado. (Lenstra e Rinnooy Kan 1978; Boctor 1990). Segundo Karp (in: Norbis e Smith 1986), no problema de programação com restrição de recursos, a forma como as restrições de precedência interagem com as restrições nos recursos, derivando para conjuntos desconexos, caracteriza o problema como combinatorial NP-completo.

De acordo com Bennington e McGinnis (1973) e Talbot (1982), pode-se assumir, sem perda de generalidade, que o problema geral de programação de projetos pode ser representado como um grafo acíclico (de atividades no nó) **G** = (N,P), onde N representa o conjunto de *atividades* e **P**, o conjunto de arcos, corresponde às *relações de precedência.* Associado a cada atividade $i \in \mathbf{N}$ ($i = 1,..,n$) existe um conjunto de possíveis *durações* d_i e as correspondentes quantidades demandadas r_{ij} do recurso tipo j ($j = 1,...,m$). A disponibilidade do recurso j no período t pode ser escrita como $a_j(t)$. Cada combinação duração-recursos é denominada *modo.* Por exemplo, suponha que é possível completar uma dada atividade *A* em 5 dias empregando uma pessoa experiente (modo 1) ou em 7 dias usando uma pessoa inexperiente (modo 2).

Uma solução viável para o problema em questão é aquela que satisfaz, em primeiro lugar a limitação dos recursos e, em segundo as restrições tecnológicas de execução das tarefas (Mohanthy e Siddiq, 1989). Neste contexto, o problema geral de programação de projetos com restrição de recursos pode ser estabelecido como: *dado um conjunto de atividades interrelacionadas (relações de precedência), onde cada atividade pode ser executada de um, entre vários modos, cada um caracterizado por uma duração e requerimento de recursos. Pergunta-se: quando cada atividade deve começar e qual modo deve ser adotado de forma a otimizar algum objetivo determinado pela administração?* (Boctor, 1990).

Classificação Dos Problemas De Programação De Projetos

A partir da literatura especializada, pode-se estabelecer que não somente existem diferentes versões para os problemas, como também há vários procedimentos propostos para uma topologia particular, as quais são caracterizadas por premissas e limitações diversas. Vários trabalhos de revisão bibliográfica relacionados ao problema em questão estão disponíveis na literatura (Herroeien 1972, Davis 1973, Bennington e McGinnis 1973, Davis 1974, Mohanty e Siddiq 1989, Boctor 1990), assim como, trabalhos que comparam desempenhos de algoritmos ótimos e/ou heurísticos (Patterson 1973, Davis e Patterson 1975, Cooper 1976, Holloway *et al.* 1979, Kurtulus e Davis 1982, Patterson 1984, Dumond e Mabert 1988, Badiru 1988, Boctor 1993, Oguz e Bala 1994)

Similarmente ao esquema proposto para classificar os problemas de programação da produção, a Figura 2.4 ilustra uma categorização do problema aqui tratado, fundamentado na descrição das premissas básicas assumidas para as atividades e os recursos durante a formulação dos modelos.

As seguintes suposições são frequentemente assumidas na formulação dos paradigmas de programação de projetos:

1. *Suposições sobre as atividades*

- Cada atividade pode acontecer somente em um intervalo de tempo ao longo do horizonte do projeto.
- Nenhuma atividade pode iniciar, a menos que todas as suas antecessoras tenham sido completadas.
- As atividades uma vez iniciadas não podem ser interrompidas.
- As durações das atividades são inteiras e conhecidas.
- As atividades devem ser executadas de um único modo $j = 1, ..., M_i$ e uma vez iniciada neste modo, não pode mais ser mudado.

2. *Suposições sobre os recursos*

- Somente um tipo de recurso renovável é requerido pelo projeto.
- A disponibilidade dos recursos é constante ao longo da realização do projeto.
- A demanda dos recursos de uma atividade é conhecida e constante ao longo da duração da mesma.
- Não é permitida a substituição entre recursos.

O relaxamento de algumas das hipóteses básicas acima citadas, leva à versões diferentes o problema de programação de projetos.

A representação esquemática da Figura 2.4 mostra que o problema em questão pode ser classificado de acordo com:

1. O tipo de problema de alocação de recursos abordado (Herroelen 1972, Davis 1973)

   - *Problema de compressão ou "Time/cost trade-off problem":* não há restrição imposta à disponibilidade dos meios. O problema consiste em reduzir o tempo de execução do projeto, adicionando recursos à determinadas atividades, de forma que a duração de cada uma delas possa ser acelerada. O objetivo é determinar a programação de menor custo.
   - *Problema de nivelamento de recursos:* este problema surge quando existem suficientes recursos para completar o projeto, mas deseja-se manter a utilização dos mesmos a uma taxa constante.
   - *Problema de alocação de recursos limitados:* o problema acontece quando a utilização dos recursos é restrita a um certo limite. Neste caso, a meta é alocar os diversos meios as atividades de forma a minimizar alguma função objetivo. (comumente minimizar a duração do projeto).

2. O número de projetos simultaneamente programados (Mohanthy e Siddiq 1989, Boctor 1990)

   - *Projeto único:* somente atividades de um mesmo projeto competem pelos recursos disponíveis.
   - *Projetos múltiplos:* a competição por recursos disponíveis é realizada entre atividades do mesmo projeto e entre projetos diferentes. Duas situações distintas podem ser identificadas: a) *situação estática* onde é assumido que todos os projetos são conhecidos inicialmente e, b) *situação dinâmica* onde novos projetos estão iniciando continuamente no tempo.

3. A condição de interrupção (Boctor 1990)

   - *Interrupção permitida:* a interrupção da execução das atividades é permitida.
   - *Sem interrupção.* uma vez começadas as atividades, não é permitido interrupções.

4. O nível de incerteza da informação do projeto

   - *Caso determinístico:* onde existe um completo conhecimento das durações e demanda de recursos das atividades.
   - *Caso estocástico:* onde assume-se durações variáveis para as atividades e/ou variação nos requerimentos dos recursos das mesmas.

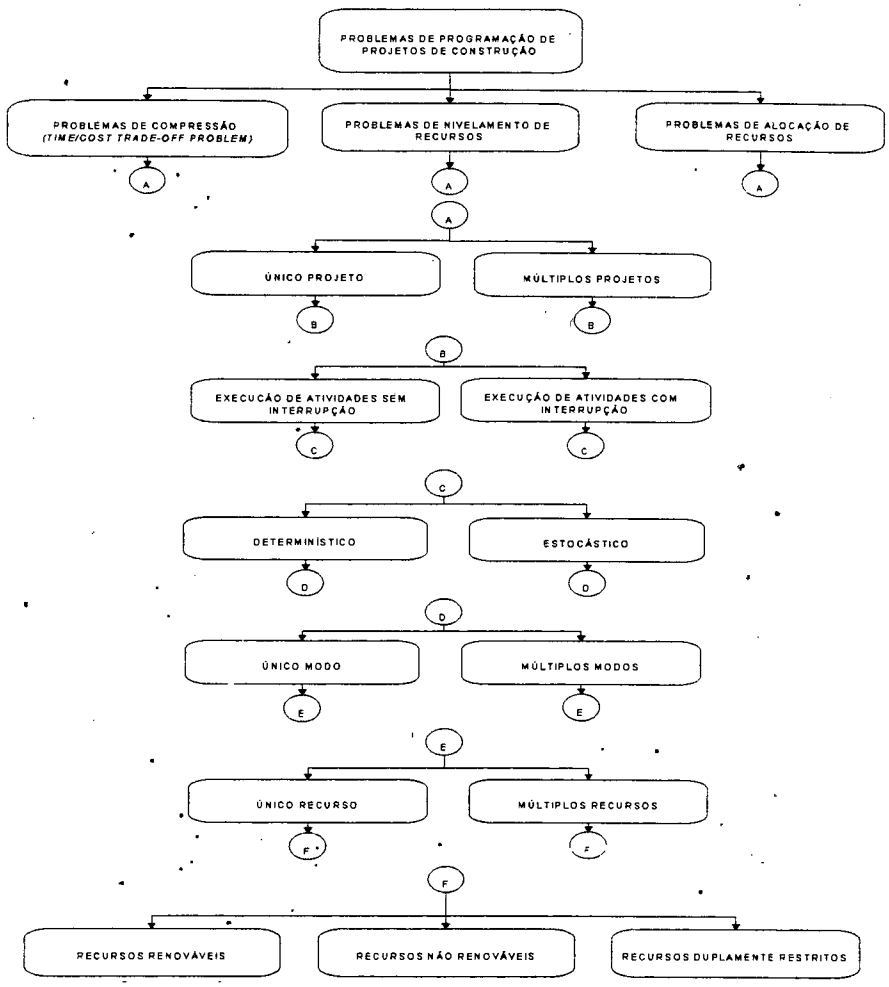


FIGURA 2.4 REPRESENTAÇÃO ESQUEMÁTICA DOS PROBLEMAS DE PROGRAMAÇÃO DE PROJETOS

5. A forma que as atividades podem ser executadas (Boctor 1990, Drexl e Gruenewald 1993)

- *Modo único:* onde cada atividade pode ser realizada em exatamente uma única forma.
- *Múltiplos modos:* quando cada atividade pode ser realizada com uma das várias formas disponíveis, isto é, as durações das atividades são funções dos recursos consumidos.

6. O número de recursos requeridos (Mohanthy e Siddiq, 1989)

- *Recurso único:* somente um tipo de recurso é requerido pelo projeto.
- *Múltiplos recursos.* Onde mais de um tipo de recurso é necessário para a realização do projeto.

7. A natureza do recurso empregado (Werglarz 1979, Slowinski 1980, Boctor 1990)

- *Renováveis:* quando os recursos estão disponíveis em quantidades limitadas a cada período, ex. mão-de-obra.

- *Não renováveis:* quando o total de consumo de recursos ao longo ou em parte do projeto é restrito, ex. dinheiro
- *Duplamente restritos:* quando a disponibilidade de recursos é limitada por períodos e para o total do projeto, ex. dinheiro.

A classificação segundo o número de critérios de otimização não foi incluída no esquema proposto, dado que a grande maioria dos trabalhos publicados utilizam uma única função objetivo, como por exemplo, *minimização da duração do projeto ou maximização da utilização dos recursos.* Outros objetivos frequentemente reportados são *minimização do custo total do projeto* e *maximização do valor presente.*

Outra suposição importante frequentemente feita, está relacionada com o tipo de relação de dependência entre atividades, ou seja, a utilização ou não de sobreposição de atividades. A grande maioria dos trabalhos publicados tratam da relação do tipo fim-início, isto é, sem sobreposição de atividades.

A seguir, como resumo do esquema proposto para a categorização dos problemas de programação de projetos, alguns trabalhos e resultados estão ilustrados na Tabela 2.7. Estas referências não tem a intenção (nem poderia, dada a extensão da literatura) de ser exaustivas.

Medidas De Desempenho De Projetos	
D_{max}	*Duração máxima do projeto.*
$\overline{D_{max}}$	*Tempo médio de realização do projeto.*
f_{max}	*Custo máximo do projeto.*
P_S	*Deterioração do projeto (project slippage).* (Ver Mohanthy e Siddiq, 1989).
E	*Eficiência da programação com restrição de recursos.* (Ver Mohanthy e Siddiq, 1989).
P_D	*Atraso total do projeto.* (Ver Mohanthy e Siddiq, 1989).
P_W	*Atraso ponderado total.* (Ver Mohanthy e Siddiq, 1989).
Rt	*Tempo total de recursos ociosos.* (Ver Mohanthy e Siddiq, 1989).
Ru	*Utilização de recursos* (Ver Khattab e Choobineh, 1991).
AT_k	*Desvio médio da melhor duração conhecida* (Ver Khattab e Choobineh, 1991).
F_D	*Frequência de obtenção da menor duração* (Ver Khattab e Choobineh, 1991).
R_R	*Amplitude de recursos* (Ver Khattab e Choobineh, 1991).

TABELA 2.7 PROBLEMAS DE PROGRAMAÇÃO DE PROJETOS

Função Objetivo	Método	Referência	Observações‡
D_{max}	Ótimo	Mandeville 1965, Carruther 1966, Burton 1967, Johnson 1967, Patton 1968, Sunaga 1970, Fisher 1970, Hasting 1972 (in: Davis 1973)	
	Ótimo	Wiest 1962, , Norden 1963, Hadlet 1964, Brand et al. 1964, Karush 1966, Burton 1967, Balas 1970, Davis 1968,	Múltiplos recursos.
	Ótimo	Davis e Heidorn 1971, Talbot 1976, Stinson et al. 1978, (in: Patterson 1984)	Múltiplos recursos.
	Ótimo	Wagner et al. 1964 (in: Davis 1973)	Demanda variável de recursos.
	Ótimo	Bell e Park, 1990	Múltiplos recursos.
	Ótimo	Tavares, 1990	
	Ótimo	Demeulemeester e Herroelen, 1992	Múltiplos recursos.
	Ótimo	Drexl e Gruenewald, 1993	Múltiplos recursos, múltiplos modos.Caso de recursos renováveis , não renováveis e duplamente restritos.
	Heurística	Kelley 1959 (in: Davis, 1973)	Múltiplos recursos.
	Heurística	Moshman et al. 1963, Butler 1961, Vershines 1963, Wiest 1969 (in: Davi s 1973)	Múltiplos projetos, múltiplos recursos.
	Heurística	Vershines, 1963, Kidd, 1963 Brand et al. 1964 (in: Davis 1973)	Múltiplos recursos.
	Heurística	Jones 1984	Múltiplos recursos.
	Heurística	Elsayed e Nasr, 1986	
	Heurística	Leachman et al. 1990	Múltiplos recursos, duração váriavei para as atividades.
	Heurística	Tsubakitani e Deckro, 1990	Múltiplos projetos, múltiplos recursos.
	Heurística	Bell e Han, 1991	Múltiplos recursos.
	Heurística - Local Search	Sampson e Weiss, 1993	Múltiplos recursos.
	Heurística	Moselhi e Lorterapong, 1993	Múltiplos recursos.
D_{max} , R_U , *manter a realização do projeto menor que a due date*	Ótimo	Davis et al. 1992	Múltiplos recursos, múltiplos objetivos.
D_{max} , $\overline{D}_{max}$	Heurística	Norbis e Smith, 1986	Múltiplos recursos.
D_{max} , *valor presente*	Heurística	Patterson et al. 1990	Múltiplos recursos, Múltiplos modos
D_{max} , *Atraso total*	Ótimo	Pritsker et al. 1969 (in: Davis, 1973)	Múltiplos projetos, múltiplos recursos; permitida interrupção das atividades.
D_{max} , f_{max}	Ótimo	Talbot, 1982	Múltiplos recursos, múltiplos modos.
f_{max}	Ótimo	Mason e Moodies 1971 (in: Davis 1973)	
otimizar trade-off do custo de atraso entre projetos	Heurística	Kim e Leachman, 1993	Múltiplos projetos, múltiplos recursos, duração variavel das atividades.
P_S , E, P_D , P_W , Rt	Ótimo e heurística	Mohanty e Siddiq, 1989	Múltiplos projetos, múltiplos recursos, múltiplos objetivos.
R_U , AT_k , R_R , P_D	Heurística	Khattab e Choobineh, 1991	

‡ A menos que seja indicado o contrário, assume-se que os problemas sejam determinísticos, não interrompiveis, sujeitos a restrição de um único recurso, único modo, recursos renováveis.

2.7.3. Problema De Linha De Montagem

O objetivo deste tipo de problema, é decidir o número mínimo de estações de trabalho que são atribuídas a elementos de trabalho com alguma relação de precedência, ou minimizar o tempo de ciclo, isto é, a quantidade máxima de tempo de processamento em cada estação, para um dado conjunto de estações de trabalho (Bellman *et al.* 1982).

De uma maneira mais formal, o problema da linha de montagem é definida como sendo o seguinte problema de decisão (ver Falkenauer e Delchambre, 1992):

Dado um grafo acíclico direcionado $G=(T,P)$ (os nós T representando as tarefas, e os arcos as relações de precedência) com uma constante L_i (extensão da tarefa) atribuída a cada nó T_i, uma constante C (o tempo de ciclo) e uma constante N, pergunta-se: podem os nós T serem divididos em N ou menos subconjuntos S_j (a j-éssima tarefa da estação) de tal forma que (1) para cada subconjunto, a soma de L_i associada com os nós no subconjunto não exceda C, e (2) existe uma ordenação do subconjunto tal que, sempre que dois nós em subconjuntos distintos são unidos por um arco em G, o arco vai do subconjunto de mais alta ordem (cedos) para um de mais baixa ordem (tardes).

Não foi aprofundado o estudo neste deste tipo de problema devido ao, relativamente fraco interesse entre a questão abordada pelo trabalho e este problema.

Embora o problema de linha de montagem e o de programação de projetos com restrição de recursos possam ser formulados de tal maneira a enfatizar as suas aparente similaridades, como Wilson e Mandeville (in: Davis 1973) sugerem, ambos diferem consideravelmente do ponto de vista administrativo. Enquanto o problema de linha de montagem está preocupado com um grande número de operações repetitivas e um grande número de produtos, o de programação de projetos geralmente é do tipo: um produto, uma operação por vez (Weist in: Davis 1973).

2.8. Relações E Principais Diferenças Entre Os Problemas De Programação

Como Davis (1973) colocou, há fortes similaridades entre alguns problemas de programação da produção, programação de projetos com restrição de recursos e linha de balanço, ao ponto que procedimentos originalmente desenvolvidos para um tipo de problema tenham sido aplicados nos outros, com considerável sucesso.

O problema de programação de projetos sujeito a restrição de recursos, tem similaridades conceituais com a programação de máquinas que tem estimulado interesses de aplicação recíproca de modelos de solução (Davis, 1973).

Contudo, "o paradigma de programação de projetos com limitação de recursos é fundamentalmente mais pragmático que a programação de máquinas, e leva em consideração muito mais das complexidades do ambiente de programação. São consideradas relações tecnológicas mais complexas e requerimento de múltiplos recursos" (Rodammer e White, 1988).

Na prática, uma das maiores diferenças entre a programação job shop e a programação de projetos, é a natureza contínua do trabalho e do fluxo no ambiente job shop e a formação de filas de produtos entre operações de processamento (Davis, 1973).

A programação de projetos e de múltiplos projetos com restrição de recursos, é considerada generalização da programação job shop tradicional e job shop com várias máquinas do mesmo tipo respectivamente (Bennington e McGinnis 1973; Bellman *et al.* 1982 e Rodammer e White 1988). Segundo Patterson *et al.* (1990), o problema de programação de projetos sujeito a restrição de recursos subjuga os problemas de programação job shop, flow shop, linha de montagem e problemas de programação análogos.

A Tabela 2.8 ilustra relações mútuas básicas entre estes três tipos de problema.

Programação De Múltiplos Projetos Com Restrição De Recursos	Programação Job Shop
Projeto	Produto
Atividade	Operação (cada produto em cada máquina)
Relação de precedência	Ordenação
Recurso	Máquina
Disponibilidade de recursos	Número de máquinas idênticas
Requerimento de recursos de cada atividade	Número de máquinas idênticas que podem processar cada operação

Programação De Um Único Projeto Com Limitação De Recursos	Linha De Montagem
Unidade do período (dias)	Estação de trabalho
Atividade	Elemento de trabalho
Relação de precedência	Relação de precedência
Quantidade máxima disponível de recursos	Tempo de ciclo
Demanda de recursos de cada atividade	Tempo de processamento de cada elemento de trabalho

TABELA 2.8 RELAÇÕES MÚTUAS ENTRE PROBLEMAS BÁSICOS DE PROGRAMAÇÃO - ADAPTADO DE DAVIS (1973)

2.9 Programação De Projetos De Construção Com Recursos Limitados

Nas seções seguintes, será discutida e formalmente definida a natureza do problema da programação de projetos com restrição de recursos. Também, as publicações consideradas mais relevantes ao desenvolvimento deste trabalho são brevemente descritas.

2.9.1 Noções Fundamentais

O principal objetivo durante um processo de construção é completar o projeto em tempo, dentro do orçamento, paralelamente a outros requisitos e especificações de qualidade estabelecidos (Rasdorf e Abudayyeh 1991).

Uma administração com sucesso de um projeto de construção, requer a elaboração de um plano mestre completo, definido no tempo, realístico e prático (Prentis 1989). Normalmente, no desenvolvimento de um projeto, todas as operações envolvidas na construção são divididas em quatro fases funcionais: planejamento, programação, monitoramento e controle, como mostra-se na Figura 2.5. Estas quatro fases são separadas em dois estágios diferentes: *estágio de preparação* (planejamento de programação), e *estágio de implementação* (monitoramento e controle).

O planejamento é aquilo que deve ser realizado no futuro para atingir os objetivos, sendo executado em vários níveis e em vários períodos de tempo ao longo do projeto.

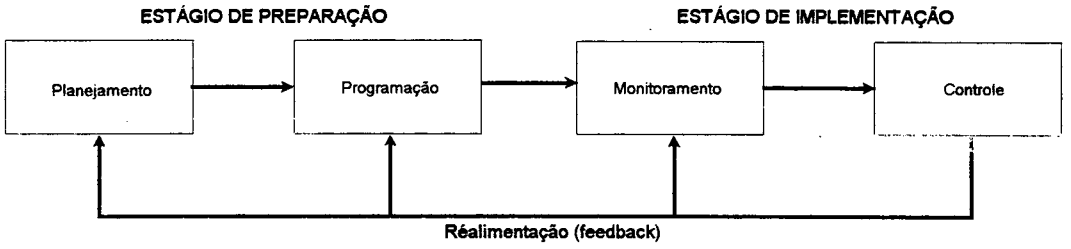


FIGURA 2.5 FASES DAS OPERAÇÕES NA ADMINISTRAÇÃO DE PROJETOS DE CONSTRUÇÃO

Na implementação do presente trabalho, o interesse maior é no nível aqui denominado *planejamento para a programação*. Este nível está relacionado com as suposições assumidas sobre os modos, relações de precedência, relações de dependência (sobreposição ou não), representação lógica (tarefas nos nós ou nos arcos) das atividades, bem como informações relativas aos recursos (demanda, disponibilidade, tipo e classe), custos e critérios de decisão.

A programação é a determinação do tempo e da sequência de execução das atividades do projeto de forma a atingir algum objetivo administrativo. A programação determina como cada atividade deve ser realizada, isto é, qual opção recurso-duração deve ser empregada e quando cada tarefa deve ser iniciada e terminada (Talbot 1982). Esta fase específica é denominada *Programação Planejada.*

O monitoramento é a identificação daquilo que atualmente está sendo realizado. É a coleta de informações sobre o desempenho no tempo do projeto e o confronto do progresso do projeto contra a programação planejada. Esta fase denomina-se *Atualização da*

Programação e diz respeito aos efeitos que as mudanças do plano de trabalho tem sobre a porção do projeto ainda não realizado. (Caliahan et al. 1992).

O controle usa os dados obtidos através do monitoramento para trazer o trabalho de volta à programação planejada. Esta parte da administração de projetos de construção é denominada *reprogramação.* A reprogramação normalmente é realizada para adaptar a programação original às situações de campo.

Finalmente, para completar o ciclo existe um quinto elemento denominado de *realimentação* (feedback) que permite fazer alterações, tanto na fase de planejamento quanto na fase de execução.

O estudo de cada uma desta fases leva à identificação de várias linhas de pesquisas, agrupadas em três grandes correntes.

Primeira Corrente De Pesquisa

A primeira corrente de estudos está focalizada no desenvolvimento de métodos e abordagens para o planejamento para a programação, anteriormente citado. Esta corrente está principalmente preocupada com a definição das relações e a representação lógica das tarefas de construção.

Segunda Corrente De Pesquisa

A segunda corrente objetiva desenvolver modelos de programação que assumem que as informações básicas de entrada como duração, dependências, modos, custos, recursos, etc. já foram determinados. Esta corrente lida com a elaboração de procedimentos que levam em consideração três principais restrições: tempo, recursos e desempenho.

Esta corrente por sua vez, pode ser subdividida em dois grandes grupos. O primeiro grupo é formado por aqueles procedimentos que tipicamente pressupõem a não restrição de recursos, que são fundamentalmente diferentes daqueles que levam em consideração a limitação dos meios. Neste contexto três tipos diferentes de problemas são identificados: os *problemas de compressão de projetos tempo/custo* e os *problemas de restrição de tempo* que formam o corpo do primeiro grupo e os *problemas de restrição de recursos* formando o segundo grupo.

Problemas De Compressão De Projetos Tempo/Custo

Os problemas tratados por este grupo são caracterizados pela suposição de que o desempenho de algumas ou de todas as atividades de um projeto pode ser melhorado alocando mais recursos a elas, a despeito de um custo direto maior (Davis 1973).

O processo de redução da programação para diminuir o tempo de realização do projeto é denominado *compressão* e os problemas deste tipo são conhecidos como *Compressão de*

Projetos Tempo/Custo (Time/Cost Trade-Off problems). Este tipo de problemas surge quando é imperativo para o administrador ajustar as durações e o sequenciamento das atividade para atender requisitos contratuais, ainda que ajustes ineficientes, como múltiplas contratação e demissões, horas extras, ou equipes grandes sejam empregados. (Callaham et al. 1992)

Os procedimentos de compressão usam uma avaliação do tempo e custo variável das atividades para determinar qual duração dever ser reduzida para minimizar economicamente a duração total do projeto, tradicionalmente sob a premissa de recursos ilimitados.

Muitos trabalhos podem ser encontrados na literatura como Elmaghraby e Pulat (1979), Moder et al. (1983), Mustafa e Murphree (1989), Johnson e Schou (1990), Callaham at al. (1992) entre outros, que abordam o problema de compressão de projetos através de algoritmos exatos e/ou heurísticos.

Elmaghraby e Pulat (1979) propõem um abordagem exata ao problema em questão levando em conta o custo de compressão do projeto, bem como penalidades para a realização de um conjunto de marcos referenciais do mesmo.

Moder, Philips e Davis (1983) discutem a implicação sócio-econômica da compressão de projetos. Eles concluem que a duração ótima de um projeto é aquela onde o custo marginal da compressão é igual aos seus benefícios marginais.

Mustafa e Murphree (1989), modelam este problema como uma metodologia de tomada de decisão de múltiplos critérios que denominaram de processo de hierarquia analítica - AHP (analytic hierarchy process). Esta abordagem permite considerar critérios subjetivos e não subjetivos durante o processo da escolha do esquema de compressão mais apropriado.

Johnson e Schou (1990), exploram regras efetivas de aceleração de projetos quando redes estocásticas são empregadas

Callaham at al. (1992), usa uma abordagem matemática baseada na relação custo-duração para tratar o problema em questão. Esta relação é representada como uma série de segmentos usados de forma a aproximar a curva para o custo de compressão e os pontos de duração normal. É a inclinação destes segmentos que é empregada para determinar o impacto sobre o custo do projeto quando a sua duração é reduzida.

Problemas Com Limitação de Tempo ou De Nivelamento de Recursos

É bastante frequente que, para muitos projetos e por razões diversas, existam datas de início e término que não podem ser mudadas. Neste contexto, são alocados tantos recursos quantos sejam necessários para assegurar a realização em tempo. Por outro lado, existem instâncias onde os custos de contratação e dispensa de mão-de-obra ou recursos materiais a curto prazo podem ser substanciais.

Neste sentido, o problema de nivelamento de recursos surge quando há recursos suficientes e o projeto deve ser concluído numa data específica, porém é desejado ou necessário reduzir a variação nos padrões de utilização dos recursos ao longo da duração do projeto. (Moder et al. 1983).

A principal hipótese assumida no desenvolvimento de qualquer procedimento para tratar este tipo de problema é que não é permitido aumentar a duração do projeto além da data pré-estabelecida. A idéia central dos procedimentos usados neste problema fundamentam-se na reprogramação das atividades dentro dos limites determinados pelas suas folgas de modo a obter uma melhor distribuição dos recursos (isto é, assume-se que existe uma programação básica que fornece o conjunto de datas de início e término das atividades)

Este tipo de problema, para o qual vários procedimentos heurísticos e exatos existem, como por exemplo Burgess e Killebrew in: Moder et al. (1983 p.205), Chang (1987), Harris (1990), Seibert e Evans (1991), Harris in: Callahan at al. (1992 p.283), para citar os mais recentes, será abordado paralelamente ao problema central do trabalho.

Problemas Com Restrição De Recursos

Segundo Moder et al. (1983), esta categoria de problemas, que é muito mais comum na indústria da construção, surgem quando há limitações na quantidade disponível dos recursos necessários à execução do projeto (ou projetos). Quando a utilização dos recursos é restrita a uma dado limite, o objetivo pode ser a alocação deles às atividades de modo a que a duração do projeto seja minimizada. Outros objetivos, como minimização dos custos, minimização do atraso, etc. não são inusuais (Davis 1973).

O problema de programação de projetos com restrição de recursos forma o escopo do presente trabalho e, neste sentido, será tratado em maiores detalhes nas seções posteriores.

Terceira Corrente De Pesquisa

A terceira corrente de pesquisa, uma extensão da segunda corrente, diz respeito à monitoração e ao controle do progresso do projeto de construção. No desenvolvimento de procedimentos para estes problemas, a maioria das vezes assume-se como entrada do sistema, que a melhor programação possível já foi determinada através de qualquer modelo. Paradigmas tradicionais de monitoramento e controle recompilam custos e durações utilizando dados atuais do desempenho de campo do projeto.

2.9.2 Tipo De Problema Abordado

O Método do Caminho Crítico (CPM) e o Program Evaluation and Review Technique (PERT) são as ferramentas mais comumente utilizadas nas últimas três décadas na administração de projetos de construção. Entretanto, é bem reconhecido que os

procedimentos PERT/CPM básicos são limitados, no sentido de não levar em consideração a disponibilidade de recursos no processo de programação. Como resultado, muitos trabalhos que tem sido desenvolvidos desde que estas técnicas foram introduzidas, corrigem esta deficiência.

Sob condições de limitação de recursos, são necessárias decisões de sequenciamento de forma a satisfazer a demanda das atividades concorrentes e a minimizar o aumento da duração do projeto além da data determinada desconsiderando esta restrição.

O problema de programar um conjunto de atividades sem que as suas relações de precedência e os limites imposto aos recursos sejam violados, não é uma tarefa fácil. Esta dificuldade aumenta quando, simultaneamente, algum critério de otimização deve ser cumprido (Moder et al. 1983).

Em geral, o objetivo maior deste tipo de problema, comumente conhecido como problema de programação com restrição de recursos, é encontrar uma programação viável que minimize a duração do projeto (Davis e Patterson 1975, Boctor 1990, Khattab e Choobineh 1991).

2.9.3 Abordagens Para O Problema

O problema de programação de projetos com restrição de recursos tem sido fonte de investigação desde os anos 60 (Ver seção 2.7.2) e um grande número de publicações e paradigmas heurísticos e exatos tem sido propostos para uma infinidade de versões do problema, assim como trabalhos de comparação entre abordagens diferentes. (Patterson 1973, Davis e Patterson 1975, Cooper 1976, Holloway *et al.* 1979, Kurtulus e Davis 1982, Patterson 1984, Elsayed and Nasr 1986, Dumond and Mabert 1988, Mohanthy e Siddiq 1989, Boctor 1990, Boctor 1993, Oguz e Bala 1994).

Entretanto, a maioria dos problemas práticos desta área são representativos dos problemas combinatoriais NP-pesados (Wiest 1967, Lenstra e Rinnooy Kan 1978, Blazewicz et al. 1983, Boctor 1990).

Devido ao crescimento exponencial do tempo computacional em função do tamanho do problema, a geração de soluções ótimas globais, ainda para problemas reais de tamanho modesto, permanece em princípio, computacionalmente impraticável através de métodos de enumeração completa. (Wiest 1967, Davis e Patterson 1975, Patterson 1984, Moder et al. 1983, Davis et al. 1992).

Neste sentido, dado que os procedimentos baseados em abordagens ótimas não tem se mostrado favoráveis, procedimentos baseados em heurísticas tem sido extensivamente utilizados como abordagens de solução ao problema em questão. Estes tipos de abordagem são hoje, as formas mais eficientes de obter uma solução boa e satisfatória para o problema,

em tempo cumputacional razoável (Davis e Patterson 1975, Moder et al. Boctor 1990, Khattab e Choobineh 1990).

A Figura 2.6 mostra o esquema para uma abordagem heurística básica para o problema aqui abordado.

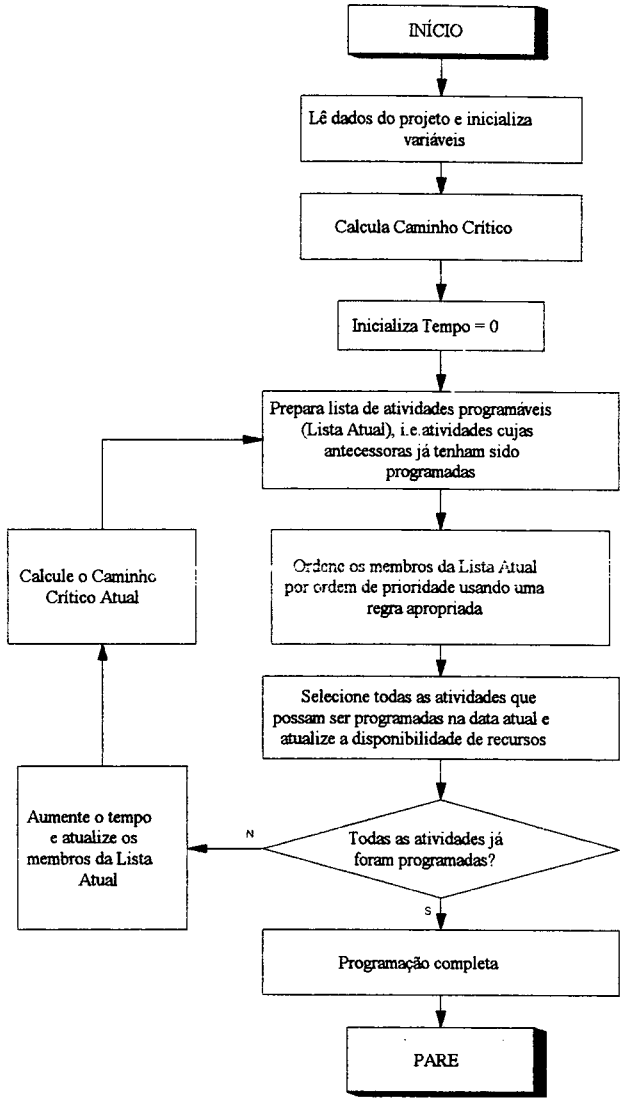


FIGURA 2.6 PROCEDIMENTO HEURÍSTICO BÁSICO PARA A PROGRAMAÇÃO DE PROJETOS

Neste contexto, na implementação do modelo proposto, uma abordagem heurística é utilizada para a solução do problema.

Tipicamente, as premissas básicas da maioria das heurísticas utilizadas na alocação de recursos, empregam regras de prioridade para selecionar quais as atividades que devem ser executadas ou retrasadas. Infinidade de regras tem sido propostas e utilizadas por pesquisadores e usuários. Em Patterson (1973), Davis e Patterson (1975), Kurtulus e Davis (1982), Dumond e Mabert (1988), Boctor (1990), Khattab e Choobineh (1991), e Boctor

(1993) podem ser encontradas as fundamentações básicas das regras mais comuns resumidas na Tabela 2.9.

REGRA DE PRIORIZAÇÃO	DESCRIÇÃO
MINSLK	(Minimum Slack First) Atividades que tem a menor folga são programadas primeiro Folga = Última Data De Início 'Primeira Data de Início
MAXSLK	(Maximum Slack First) Selecionar primeiro atividades de maior folga.
SDF	(Shortest Duration First). Programar primeiro as atividades com menor duração.
LDF	(Longest Duration First). Selecionar as atividades de maior duração
SASP	(Shortest Activity from Shortest Project). Programar primeiro atividades do projeto de menor duração (múltiplos projetos).
LALP	(Longest Activity from Longest Project). Programar primeiro atividades do projeto de maior duração (múltiplos projetos)
MINLFT	(Minimum Late Finish Time). Priorizar atividades com menor Última Data de Término.
GTRD	(Greatest Total Resource Demand). Priorizar atividades que necessitem o maior número de recursos de todos os tipos.
SRD	(Smallest Resource Demand). O inverso da regra GTRD
GRU	(Greatest Resource Utilization). Esta regra prioriza a combinação de atividades que resulte na máxima utilização dos recursos a cada intervalo
RSM	(Resource Scheduling Method). Para tempo *t* calcular para cada par de atividades programáveis *i* e *j* o índice x_{ij} = max(0, *t*+duração da *i-éssima* Última Data de Início de *j*). Schedule first the activity having the smallest x_{ij}.
MAP	(Most Activities Possible). Esta regra dá prioridade à combinação de atividades que resulte no maior número de atividades programadas em qualquer intervalo de tempo.
MAXNIS	(Maximum Number of Immediate Sucessors). Escolhe primeiro a atividade que tenha o maior número de arcos posteriores.
MAXCAN	(Maximum Number of Subsequent Candidates). Prioriza as atividades que tem o maior número de atividades candidatas subsequentes. Atividades candidatas subsequentes são aquelas que se tornam ou permanecem programáveis se a atividade em questão iniciar no período em consideração.
MAXRWK	(Maximum Remaining Work). Escolhe primeiro a atividade que tenha a maior soma possível das durações da atividade e todas as suas sucessoras.
MINRWK	(Minimum Reamining Work). Escolhe primeiro a atividade que tenha a menor soma possível das durações da atividade e todas as suas sucessoras.
FCFS	(First Come Firts Served) Dá maior prioridade a primeira atividade programável que aparecer.
ACTIM	(Activity-Time). Computa-se para cada atividade a diferença entre a Última Data e a duração do Caminho Critico
ACTRES	(Activity-Resource). Esta é a combinaçao da duração da atividade e a demanda dos recursos, multiplicando ambos elementos
TIMRES	(Time-Resource). É uma regra composta de porções ponderadas das regras ACTIM e ACTRES.
GENRES	É uma modificação da regra TIMRES dando pesos diferentes para ACTIM e ACTRES.
RAN	(Randomness) Seleção aleatória das atividades. Esta regra nunca é aplicada sozinha.

TABELA 2.9 REGRAS HEURÍSTICAS MAIS COMUNS

2.9.4 Formulação Geral Do Problema Abordado

O problema de programação de projetos com limitação de recursos pode ser formalmente caracterizado pelas seguintes premissas (Slowinski 1981, Drexel e Gruenewald 1993):

- Um projeto composto de um conjunto **A** de *n* atividades, $\{A_1, ..., A_n\}$.
- As atividades não podem ser interrompidas durante a sua execução.

- Um conjunto de critérios de otimalidade **Q** para o desempenho do conjunto **A** composto de: (1) *critério de tempo*, como minimização da duração do projeto, $T = \min [t_i]$, onde t_i é o tempo de realização de A_i; a minimização do atraso $L = \min[t_i - d_i]$, onde d_i é a data (due date) de realização ideal de A_i, e (2) *critérios de custo.*

- A atividade A_i pode ser executada em um dos diferentes modos $m = 1, 2, ..., M$. Toda atividade, uma vez iniciada num determinado modo, deve ser concluída sem possibilidade de mudança.

Programar a atividade A_i no modo m implica numa duração d_{im}. Um conjunto **R** de recursos renováveis $\{R_1^R, ..., R_p^R\}$ classificados em p tipos, cada um disponível em B_{kt}^R unidades no período t, $k = 1,2, ..., p$. Programar uma atividade A_i no modo m, usa r_{imk}^R unidades por período do recurso R_k^R.

Baseado nas premissas acima citadas, o problema de programação de projetos com restrição de recursos é definido da seguinte maneira:

Definição (Sampson e Weiss 1993, Demeulemeester e Herroelen 1992): Determinar as datas de início das atividades de um projeto que maximize alguma determinada função objetivo, sujeito à restrições de precedência e alguma definição determinística viável.

$$\text{maximizar } f(T) \quad f \in \mathbf{Q} \tag{2.1}$$

sujeito a:

restrições de precedência

$$f_j - f_i \geq d_{jm} \qquad (i, j) \in \mathbf{H}, \mathbf{H} \subset \mathbf{A}, \tag{2.2}$$

disponibilidade de recursos

$$\sum_{S_t} r_{im}^R \leq B_k^R \quad r \in \mathbf{R}, \tag{3.3}$$

$$m = 1, ..., M$$

$$k = 1,2, ..., K\ (K = p)$$

onde:

f_i = data de término da atividade i, $i \in \mathbf{A}$,

H = conjunto de pares de atividades indicando restrições de precedência,

d_{im} = tempo de processamento da atividade i no modo m,

r_{imk}^R = quantidade do recurso renovável e tipo k, requerido pela atividade i no modo m.

S_t = conjunto de atividades em execução no intervalo $]t - 1, t]$,

B_k^R = disponibilidade total do recurso renovável do tipo k.

2.9.5 Trabalhos Prévios

A seguir é apresentada uma revisão sucinta da literatura, considerada mais relevante ao desenvolvimento do trabalho.

Como já foi ressaltado em seções anteriores, o problema de programação de projetos com restrição de recursos é relativamente um dos que maior atenção tem recebido na literatura. Porém, a maioria dos algoritmos atualmente disponíveis não atendem completamente as situações reais, devido principalmente, às limitações impostas nas premissas básicas sob as quais eles são desenvolvidos. Diferentes maneiras de abordar o problema leva a diferentes versões para o problema em questão.

Medidas de desempenho para os projetos

A maioria das versões estudadas na literatura utilizam a minimização da duração do projeto como critério de otimalidade, entretanto várias abordagens tratam outros objetivos ou múltiplos objetivos diferentes, como por exemplo Talbot (1982), Deckro e Hebert (1989), Patterson et al. (1990), Sampson e Weiss (1993), Rascoe et al. (1992).

Talbot (1982), introduz métodos para formular e resolver o problema de programação de projetos onde a duração é função dos recursos. Ele desenvolveu uma metodologia de dois estágios que usa regras heurísticas no primeiro estágio e um algoritmo de enumeração implícita no segundo estágio, objetivando minimizar a duração do projeto. Este mesmo algoritmo básico é utilizado para abordar o problema de compressão de projetos com limitação de recursos.

Deckro e Herbert (1989), desenvolveram modelos ótimos para dois casos específicos, a compressão de recursos críticos e a compressão da duração das atividades, estendendo modelos de programação tradicionais para tratar a compressão de projetos com restrição de recursos. O primeiro modelo desenvolvido está baseado no modelo proposto por Pritsker, Watters e Wolfe (1969). O algoritmo que trata do segundo caso está fundamentado no modelo de Bowman (1959).

Patterson et al. (1990), reporta uma experiência computacional para resolver a versão de múltiplos modos do problema de programação com restrição de recursos para as funções objetivos de minimização da duração do projeto e maximização do valor presente. Um procedimento de busca em profundidade baseada no algoritmo de branch and bound é utilizado como metodologia de solução. Este modelo consiste de duas fases: inicialização do problema de enumeração e geração do espaço de solução. Na fase de inicialização é realizada a priorização heurística das atividades e a seleção do modo de execução das atividades. Este procedimento guia a busca da solução inicial, afetando a ordem na qual as atividades e os modos serão considerados na atribuição dos recursos durante a fase de enumeração. Esta segunda fase é utilizada para gerar o espaço de solução e para avaliar,

implícita ou explicitamente, todas as soluções parciais que podem ser estendidas para uma solução ótima.

Sampson e Weiss (1993), apresentam um algoritmo de busca local para a minimização da duração do projeto sob restrição de recursos. Esta heurística é baseada na definição da solução, numa estrutura de vizinhança, que indica como mover-se de uma solução para outra possivelmente melhor, e numa condição de parada.

Davis et al. (1992), finalmente, abordam o caso de múltiplos objetivos para progamação de projetos com limitação de recursos. Eles empregam um método de programação linear multi-objetivos para desenvolver um procedimento de decisão que permita ao decisor avaliar a relação entre vários objetivos e determinar a programação de melhor preferência. O primeiro objetivo do modelo é a minimização da duração do projeto. Os objetivos remanescentes, minimizam a excessiva utilização dos vários recursos individuais ao longo do horizonte de planejamento.

O caso de interrupção de atividades

Slowinski (1981), Leachman e Kim (1992), entre outros, tratam o caso onde a interrupção e/ou a sobreposição de atividades é permitida.

As duas abordagens apresentadas por Slowinski (1981), baseadas na programação linear multi-objetivos, tratam do caso de múltiplos modos com utilização de diferentes categorias de recursos (renováveis, não renováveis e duplamente restritos) e múltiplos critérios de otimização, levando em consideração a interrupção de atividades.

Leachman e Kim (1992) por outro lado, propõem uma abordagem para lidar com relações que consideram sobreposição entre as atividades, baseada na medição do progresso das atividades em termo da aplicação acumuiativa dos recursos, ao invés do modelo tradicional baseado na duração. O modelo leva ainda em consideração durações variáveis para as atividades.

O caso múltiplos modos

São poucos os trabalhos que abordam o caso de múltiplos modos. Além dos trabalhos já citados de Slowinski (1981) e Talbot (1982), esforços recentes vem sendo realizados para formular e resolver este tipo de problema, entre os quais podem ser citados Boctor (1990) e Drexel e Gruenewald (1993).

Boctor (1990) desenvolveu um esquema geral para resolver problemas de tamanho grande, para os quais a duração das atividades é função dos recursos alocados a sua execução, objetivando minimizar a duração total do projeto. Foi realizado um estudo comparativo do desempenho de 21 regras heurísticas. O modelo emprega os conceitos tradicionais de priorização de atividades e seleção de modos.

Drexel e Gruenewald (1993) apresentam um método de programação estocástico para formular e resolver o caso de programação com múltiplos modos e restrição de recursos, bem como a compressão de projetos com múltiplos modos. A contribuição do modelo é generalizar regras de programação determinísticas apropriadas incorporando uma técnica de seleção aleatória ponderada.

Outros modelos interessantes para o problema abordado, são os desenvolvidos por Jones (1984), Li e Willis (1992) e Moselhi and Lorterapong (1993).

Jones (1992) usou uma abordagem de Monte Carlo para aleatoriamente gerar um grande conjunto de programações viáveis para então selecionar a de melhor desempenho.

Li e Willis (1992) apresentam um procedimento heurístico onde um projeto é programado do início para o fim e do fim para o início iterativamente até que não sejam obtidas melhoras na duração do projeto. Esta programação iterativa aborda diferentes categorias e tipos de recursos, com diferentes níveis de utilização, tais como: recursos renováveis constantes no tempo, recursos renováveis variáveis no tempo, recursos não renováveis e duplamente restritos constantes no tempo, recursos não renováveis e duplamente restritos variáveis no tempo.

Moselhi e Lorterapong (1993) desenvolveram um algoritmo heurístico que objetiva incrementar a consistência do desempenho da tradicional regra da folga total mínima. O modelo é baseado na alocação de recursos por grupo. Os conjuntos de atividades são geradas considerando primeiramente todas as possíveis combinações das atividades ativas em conflito e então reduzindo para aquelas viáveis, priorizadas pelo menor impacto sobre a duração do projeto.

Múltiplos projetos

Todas as abordagens acima mencionadas tratam com a alocação de múltiplos recursos para um único projeto. Entretanto, o impacto na programação quando se considera múltiplos projetos é muito maior. Entre os esforços mais recentes neste campo podem ser citados os trabalhos de Dumond e Mabert (1988), Mohanty e Siddiq (1989), Tsubakitani e Deckros (1990), Kim e Leachman (1993).

Dumond e Mabert (1988), abordam o problema de estabelecer datas de entrega num ambiente onde os projetos chegam constante e aleatoriamente no tempo. Cinco regras heurísticas e quatro procedimentos de atribuição de *due dates* foram testados usando um modelo de simulação de duas fases. Na primeira fase as regras heurísticas controlam a alocação dos recursos. Na segunda fase é demostrado o desempenho dos procedimentos de atribuição de "due dates".

Mohanty e Siddiq (1989) fizeram uma análise do problema de múltiplos projetos e restrição de recursos com due dates individuais. A análise é executada usando Programação Inteira por Objetivos - IGP (Integer Goal Programming) e simulação. O IGP é usado para

gerar as programações. O modelo de simulação é empregado para testar algumas regras heurísticas em diferentes situações de limitação de recursos. Finalmente o modelo compara as programações geradas por ambos modos e decide pela melhor opção.

Um modelo de programação e controle foi desenvolvido por Tsubakitani e Deckros (1990) para o problema em questão. O paradigma, desenvolvido a partir de dados reais de uma firma, usa as abordagens propostas por Kurtulus e Davis (1982) para a seleção das regras heurísticas apropriadas.

Kim e Leachman (1993), consideram o problema de programação de multi-projetos com custos de atraso. Eles desenvolveram um algoritmo heurístico baseado no conceito dos perfis e objetivos acumulativos dos recursos para minimizar o custo total de atraso.

Herroelen (1972) apresenta estudos anteriores sobre o tema, tais como os desenvolvidos por McGee e Markarian, Wiest, Combe, Frere et al., Oshima, Fendley, e traz um comentário sobre a técnica de RAMPS.

Monitoração e controle

Como já salientado anteriormente, são dois os estágios envolvidos no horizonte de realização de um projeto, o estágio de preparação e o estágio de implementação.

A grande maioria dos modelos acima citados são desenvolvidos para resolver o problema de programação no estágio de preparação. As decisões nesta etapa são feitas baseadas em suposições e julgamentos a priori da disponibilidade e demanda dos recursos.

Contudo na prática, na fase de execução do projeto ao tempo de construção por várias razões operacionais, a disponibilidade de recursos pode não ser suficiente para realizar todas as atividades programadas na fase de planejamento. Sob estas circunstâncias, decisões de campo devem ser tomadas sobre quais atividades devem ser executadas e quais adiadas.

Por outro lado, um modelo efetivo deve incorporar a variabilidade do projeto devido as situações de campo, desempenho da mão-de-obra, condições dos equipamentos, fornecimento de material, condições climatológicas e desempenho administrativo. Segundo Chang (1987), a maior razão para o espaço existente entre a aplicação e a pesquisa, além da dificuldade inerente à natureza combinatorial do problema, é que a maioria dos modelos desenvolvidos não possuem a capacidade de resolver o impacto das várias forças externas, como chuva, atrasos de fornecimento, disponibilidade de recursos, etc. que são inevitáveis nos projetos reais.

Recentemente, várias abordagens para a programação de projetos levam em consideração as incertezas das condições de canteiro, como Levitt e Kunz 1985, Chang 1987, 1990, Mosheli e Nicholas 1990, Hassan e Ayyub 1993, Wu e Hadipriono 1994. Muitos destes modelos são desenvolvidos usando ferramentas de inteligência artificial como os

Sistemas Especialistas. Para uma revisão mais detalhada sobre Sistemas Especialistas na administração da construção referir-se a Kostem e Maher (1986), Ashley e Levitt (1987), Quinlan (1987), Adeii (1988), Pham and Pham (1988), Formoso e Brandon (1989), Minkarah e Ahmad (1989), Mohan (1990), CIB-90 (1990), Noronha e Sarma (1991).

Levitt e Kunz (1985) desenvolveram um protótipo baseado em conhecimento que permite programar as atividades a medida que o projeto avança, baseado nos dados e o conhecimento do término das atividades. O modelo recopila a duração do projeto e o caminho crítico substituindo as durações esperadas originalmente para as atividades completadas pelas durações atuais, fornecidas pelo usuário. As durações das atividades restantes são substituídas por valores revisados. Este valores são estimados usando regras heurísticas e representam os efeitos potenciais que os fatores de risco tem sobre as durações e o consumo de recursos das atividades a serem executadas.

Chang (1987), (1990), apresenta uma abordagem de Inteligência Artificial para resolver o problema de prioridade que surge na alocação de recursos em projetos de construção. O modelo por ele proposto, usa as saídas de um sistema especialista, com regras de inferência difusas, para resolver o problema de ordenação.

Um modelo de simulação Monte Carlo é usado por Mosheli e Nicholas (1990) para abordar o problema de progresso e custo de projetos de construção sob incerteza. O modelo usa os dados obtidos no avanço do projeto para prever as durações e os custos das atividades remanescentes, bem como para atualizar a ordem de prioridade de execução das atividades ainda não iniciadas.

Hassan e Ayyub (1993), propõem uma estratégia de controle de atividades baseado em conhecimento com auto-aprendizado difuso que, segundo eles, pode permitir detectar e corrigir variáveis que podem ser reponsáveis por falhas futuras.

Um modelo desenvolvido por Wu e Hadipriono (1994) emprega conceitos de lógica difusa angular para avaliar os impactos de diferentes fatores (como condições de canteiro, desempenho de equipamentos e mão-de-obra, etc) sobre a duração da atividade. O sistema utiliza um software comercial para calcular a programação inicial, e mais tarde, para reprogramar as atividades que tenham os valores das suas durações sido modificadas pelo modelo.

2.9.6 Conclusões Da Revisão Da Literatura De Programação De Projetos

Como conclusão da breve revisão bibliográfica acima apresentada, o seguinte pode ser estabelecido:

1. Nenhum dos modelos estudados, aborda a alocação de recursos limitados e o nivelamento deles ao mesmo tempo. Ambos os problemas são tratados em

separado. A razão principal para isto, talvez sejam as premissas antagônicas sob as quais eles são fundamentados, pois, se para o problema com limitação de recursos é permitido estender a duração do projeto isto não é aceitável no nivelamento de recursos. Entretanto, pode ser desvantajoso obter programações ótimas, desde o ponto de vista da duração do projeto, a expensas de custos excessivos causados por uma má utilização dos recursos. Por outro lado, muitas vezes é irreal assumir que ao considerar o nivelamento dos recursos, estes são ilimitados. Sempre é possível assumir que, a partir de um certo nível, o custo de obtenção de recursos torna-se infinito, logo estaria-se limitando de alguma forma a sua utilização.

O modelo multi-objetivo proposto por Davis *et al.* (1992) citado anteriormente, embora considere o nivelamento de recursos em cenários com limitação de recursos, ainda trata o problema da maneira tradicional, ou seja, admite que a restrição dos recursos pode ser relaxada para permitir acelerar atividades. Neste sentido, assume-se que recursos podem ser obtidos até um limite onde o excesso na utilização de cada um deles seja mínima e que reduza a duração do projeto a uma data administrativamente preferível, menor que a ótima calculada considerando as restrições originais.

- Os modelos pesquisados, ao executar uma programação, não levam em consideração a importância relativa dos diferentes tipos de recursos. Tradicionalmente, assume-se que todos os recursos tem igual importância, logo tem a mesma influência na programação resultante.

- Nenhum dos paradigmas desenvolvidos para o problema de programação de projetos com restrição de recursos, estabelece a importância, ou não, de considerar a eficiência da utilização dos recursos como um fator que afeta a qualidade da programação obtida. Comumente os procedimentos desenvolvidos, produzem soluções viáveis gerando sistematicamente o espaço de soluções, avaliando uma função objetivo e escolhendo a melhor. Entretanto, durante qualquer processo de busca, é possível encontrar ordenações diferentes para as atividades que resultem num valor igual para a duração do projeto. Estas soluções são consideradas programações equivalentes em relação a este objetivo, porém, eventualmente com diferentes perfis de utilização de recursos.

Neste sentido, propõe-se um modelo que tenta preencher, na medida do possível, estas lacunas. O modelo heurístico desenvolvido, incorpora características que permitem resolver o problema de programação de projetos com restrição de recursos e múltiplos modos, ao mesmo tempo que a utilização dos recursos empregados é nivelada.

2.10 Conceitos Fundamentais De Algoritmos Genéticos

Segundo Reeves (1993), desde a perspectiva da Pesquisa Operacional, a idéia de um algoritmo genético pode ser entendida como a exploração inteligente de uma busca aleatória. Os Algoritmos Genéticos (AG) são uma heurística de busca e técnicas de otimização que imitam a seleção natural e o processo evolutivo biológico. Os AG foram inicialmente desenvolvidos por Holland nas décadas de 60 e 70 e formalmente introduzidos no seu livro *Adaptation in Natural and Artificial Systems* (Holland 1975). Outros trabalhos pioneiros que complementam estas idéias são os desenvolvidos por De Jong (1975) e Goldberg (1989).

Basicamente, os algoritmos genéticos empregam uma solução candidata (um ponto no espaço de busca) propriamente codificada, denominada *cromossomo*. Estes cromossomos são agrupados em conjuntos denominados *populações*. Uma população definida num dado intervalo de tempo é denominada uma *geração*. Neste contexto, os algoritmos genéticos funcionam através da combinação de pedaços de soluções de uma população, cujas aptidões são determinadas por uma apropriada *função de avaliação*, de modo que soluções melhores sejam obtidas a cada geração. A manipulação de soluções é feita por elementos, denominados *operadores genéticos*.

Os algoritmos genéticos diferem das técnicas tradicionais em duas características importantes:

1. a propriedade de paralelismo implícito, que permite analisar e avaliar um conjunto de soluções simultaneamente e não somente, estimar e melhorar uma única solução, e
2. a utilização da codificação de um conjunto de parâmetros ao invés dos próprios parâmetros.

A essas duas características, pode ser adicionado a propriedade de utilização de regras de transição probabilísticas.

Essas propriedades fornecem aos algoritmos genéticos a habilidade de mitigar o problema de mínimos locais e ainda, tratar com sucesso problemas combinatoriais NP-pesados.

A estrutura básica do funcionamento dos algoritmos genéticos pode ser descrita pela seguinte representação em pseudocódigo:

Procedimento ALGORITMO GENÉTICO
começa
Inicializa população *P(0);*
Avalia *P(0);*
t = 1;
repete
Seleciona *P(t)* de *P(t-1);*
Recombina *P(t)*;
Avalia *P(t)*;
até que a condição de parada seja verdadeira.
termina

A construção de um algoritmo genético para qualquer problema pode ser separado nas seguintes tarefas (Chan e Tansri, 1994):

1. determinação da estrutura de representação,
2. determinação da função de avaliação,
3. determinação do tamanho da população e número de gerações, e
4. determinação dos operadores genéticos e suas probabilidades associadas

Essas tarefas, bem como o conceito *esquema,* são brevemente descritos a seguir. Ver Goldberg (1989) para uma discussão mais profunda destes conceitos.

2.10.1 Esquema

O principal conceito da teoria dos algoritmos genéticos são os esquemas, os quais podem ser imaginados como a definição de subconjuntos de cromossomos similares, ou *hiperplanos* no espaço *n*-dimensional (Reeves, 1993).

Por outro lado, os esquemas contém pontos com valores robustos de aptidão, onde cada ponto (cromossomo) representa uma amostra de numerosos hiperplanos que o contém. O problema é que os hiperplanos corretos, isto é, aqueles que contém boas soluções, não são conhecidos durante a busca, porque senão a solução ótima poderia ser conhecida de antemão. Portanto, os bons hiperplanos devem ser *adivinhados* a partir das soluções mais aptas, dadas pela amostragem de esquemas durante a busca. (Falkenauer e Bouffouix, 1991).

Os detalhes dos fundamentos matemáticos, comportamentos e principais conclusões sobre os esquemas podem ser encontrados em Holland (1975). Goldberg (1989) e Reeves (1993).

2.10.2 Representação Da Estrutura De Solução

Para resolver qualquer problema de otimização usando algoritmos genéticos, é necessário a codificação dos parâmetros do problema numa estrutura que permita representar, de maneira apropriada, as soluções do problema.

A representação depende do problema abordado e não é única, no sentido de que existem muitas formas de codificar soluções. Porém uma vez definida a estrutura mais apropriada para representar a solução do problema, esta codificação deve conter todos os nós de busca e ser única (não empregar mais do que uma).

Tradicionalmente, dois tipos de estruturas de codificação são empregadas:

1. *Codificação binária:* que representa a solução do problema como uma cadeia de caracteres composta de *0* e *1*.
2. *Codificação de ordem:* onde a solução do problema é representada como uma cadeia de caracteres que guardam uma relação de ordem entre si.

2.10.3 Tamanho Da População E Número De Gerações

Como já salientado, uma das vantagens dos algoritmos genéticos sobre as técnicas tradicionais de otimização é a busca em paralelo de vários nós definidos no espaço de busca. O tamanho da busca em paralelo é denominado *tamanho da população*, que é igual ao número de representações de soluções do problema a cada geração. O tamanho da população, por ser um problema dependente, necessita ser determinado experimentalmente. Em princípio, se usar populações pequenas corre-se o sério risco de se obter soluções sub-ótimas, enquanto se usar populações grandes incorre-se em severas penalidades computacionais.

Por outro lado, o algoritmo evolui para soluções cada vez melhores, convergindo para algum ponto ótimo após um certo número de gerações. O número de gerações usadas para um problema também é um elemento que necessita ser determinado experimentalmente.

2.10.4 Operadores Genéticos

O papel dos operadores genéticos é criar novos nós de busca no espaço de soluções baseado nos elementos da população atual. Três tipos de operadores podem ser identificados: reprodução, cruzamento e mutação.

Reprodução

A reprodução é um processo de seleção empregado para determinar quais os cromossomos (elementos de uma população) gerarão descendentes que formarão parte da próxima população. Elementos com alto valor de aptidão terão maiores chances de ser selecionados do que aqueles de valores baixos. Portanto, analogamente à seleção biológica natural, os fortes continuarão a viver, os fracos morrerão.

Cruzamento

O operador de cruzamento é considerado o mais importante dos algoritmos genéticos. Ele permite a produção de novas estruturas através do intercâmbio parcial de informações (genes) entre pares de elementos.

Na literatura da área, vários operadores de cruzamento tem sido propostos, entre os quais, três são objetos de maior destaque devido a quantidade de aplicações em que são empregados.

Operador OX (Order Crossover). Este operador começa pela escolha aleatória de dois pontos de corte em cada um dos elementos selecionados. A seção definida entre estes dois pontos é copiada integralmente no descendente. Os lugares restantes são preenchidos usando as informações não repetidas na seção de cruzamento, começando do segundo ponto de corte. Por exemplo, considere os seguintes cromossomos:

Pai 1

h	k	c	e	f	d	b	l	a	i	g	j

Pai 2

a	b	c	d	e	f	g	h	i	j	k	l

que gerarão os seguinte descendentes:

Filho 1

d	e	f	g	h	i	b	l	a	j	k	c

Filho 2

e	f	d	b	l	a	g	h	i	j	k	c

A sequência *j, k, c* (volta ao início) *d, e, f, g, h, i*, é o gene contido no segundo pai, começando no segundo ponto de corte. De maneira similar, obtem-se a seqência *j, k, c, e, f, d, b, l, a,* do segundo filho.

Operador PMX (Partially Mapped Crossover). Este operador também é executado escolhendo aleatoriamente dois pontos de corte. O processo pode ser ilustrado utilizando os mesmos cromossomos do exemplo anterior.

Pai 1

h	k	c	e	f	d	b	l	a	i	g	j

Pai 2

a	b	c	d	e	f	g	h	i	j	k	l

As informações contidas entre estes dois pontos, |*b, l, a* | e |*g,h,i* |, são intercambiadas obtendo-se as seguintes representações:

Filho 1'

a	b	c	d	e	f	b	l	a	j	k	l

Filho 2'

h	k	c	e	f	d	g	h	i	i	g	j

Porém, estas duas representações intermediárias são estruturas ilegais porque possuem informações repetidas e algumas outras perdidas. Assim, o passo final é substituir estes genes repetidos mapeando (*g, h, i*) para (*b, l, a*) e vice-versa, obtendo as seguintes duas novas estruturas:

Filho 1

i	g	c	d	e	f	b	l	a	j	k	h

Filho 2

l	k	c	e	f	d	g	h	i	a	b	j

Operador CX (Cycle Crossover). O esquema do operador CX é muito diferente dos esquemas OX e PMX mostrados anteriormente. O operador CX executa recombinações de forma que cada gene dos descendentes venham da posição correspondente de qualquer um dos pais.

Dado que este é o operador a ser utilizado na implementação do modelo proposto, o mesmo será mais profundamente discutido na seção 3.3.5 do capítulo seguinte.

Resultados teóricos e empíricos comparando o desempenho dos operadores OX, PMX e CX podem ser encontrados em Goldberg (1989), Oliver *et al.* (1987) e Chan e Tansri (1994).

Mutação

O papel principal do operador de mutação, é mudar aleatoriamente parte das informações codificadas de um cromossomo para criar novos elementos. Por outro lado, a operação de mutação pode ser empregada para reordenar codificações ilegais, obtendo novas soluções viáveis e legais.

2.10.5 Aplicações De Algoritmos Genéticos À Problemas De Programação

Os algoritmos genéticos desde a introdução dos conceitos básicos realizados por Holland vem sendo aplicados a diversas áreas de pesquisa e a situações do mundo real com ótimos resultados.

Segundo Blanchard (1994), a última conferência WCCI'94 (World Congress on Computational Intelligence) acontecida em Orlando FL, USA, apresentou uma série de soluções promissoras a situações reais usando algoritmos genéticos. Blanchard relata o caso da US West, uma companhia regional de telecomunicações do estado do Colorado, que vem utilizando um sistema baseado em AG que permite projetar, em duas horas, redes óticas especializadas, tarefa que levaria seis meses usando especialistas humanos. O sistema produz soluções 10% melhores que os produzidos pelo homem. A companhia estima que o sistema permitirá uma economia de 100 milhões de dólares até o final do século.

AIS (Barcelona, Espanha) usou um sistema baseado em AG e sistemas especialistas para programar os Jogos Paralímpicos de 1992. Enquanto que nas Olimpíadas os atletas são agrupados em duas grandes classes, masculino e feminino, os competidores Paralímpicos são classificados em mais de 100 classes, de acordo às restrições médicas.

Um sistema em desenvolvimento na New Mexico State University deriva imagens faciais de criminosos a partir de testemunhas do crime, usando AG. O sistema tem se mostrado mais efetivo na produção de imagens acuradas de criminosos do que qualquer outra técnica de obtenção de informação de imagens.

No campo de programação de tarefas e de máquina, as aplicações vem crescendo e apresentando um desempenho muito satisfatório. A seguir, algumas publicações mais recentes e relevantes ao desenvolvimento do presente trabalho são brevemente apresentados.

Sponsler (1989), apresenta um sistema protótipo desenvolvido para avaliar a aplicabilidade dos algoritmos genéticos na otimização da programação do telescópio espacial Hubble. Vários operadores genéticos foram testados e o melhor AG foi comparado

com um otimizador baseado em Redes Neuronais (RN). Neste caso específico os AG não se mostraram tão eficientes quanto as RN.

Syswerda e Palmucci (1991) reportam a construção de um otimizador para uma aplicação prática de progamação de recursos no laboratório SITS (System Integration Test Station Laboratory) da Marinha Americana para o desenvolvimento do jato F-14.

Falkenauer e Bouffouix (1991), introduzem a técnica de AG para tratar o problema de job shop, apresentando uma codificação apropriada para a questão, e demonstrando o seu desempenho com exemplos de tamanhos como encontrados na realidade.

Lawton (1992), apresenta idéias muito gerais de como usar os algoritmos genéticos ao problema de programação, com tendência para a programação de projeto.

Gupta *et al.* (1993) abordam o problema n-produtos, uma máquina com o objetivo de minimizar a variância do tempo de fluxo.

Bruns (1993), apresenta um algoritmo genético avançado para a solução de problemas reais de produção. A abordagem é baseada no incremento dos AG com conhecimentos do domínio específico.

Fang, Ross e Corne (1993), descrevem uma abordagem de AG que produz resultados, para problemas de programação job-shop padrão, melhores do que esforços prévios usando esta técnica no mesmo problema. A abordagem também é viável de ser aplicada para tratar o problema de programação do tipo open-shop e reprogramação job-shop.

Gauthier (1993), desenvolveu uma abordagem baseada em AG para o problema de programação da produção em ambiente de flow-shop. Introduz um procedimento de calibração dos parâmetros do algoritmo utilizando raciocínio difuso e sistemas especialistas.

Kim e Lee (1994), desenvolveram uma metodologia para o problema de programação em ambiente job-shop baseada na utilização de AG. A metodologia é empregada como um procedimento iterativo de melhoramento de programações. O algoritmo mostrou-se eficiente para várias instâncias do problema, chegando ao ótimo diversas vezes em tempo computacional razoável.

Finalmente, cabe salientar que ainda é incipiente a pesquisa sobre a aplicabilidade dos algoritmos genéticos ao problema de programação de projetos em geral e, particularmente, muito escassa (insignificante) na área de projetos de construção.

CAPÍTULO 3

UM ALGORITMO EVOLUTIVO PARA A PROGRAMAÇÃO DE PROJETOS

Este capítulo introduz o modelo baseado em Algoritmos Genéticos proposto para resolver o problema de programação de projetos sob restrição de múltiplos recursos, no qual a duração de cada atividade depende da quantidade de recursos alocados para a sua execução, isto é, cada atividade pode ser realizada num dos vários diferentes modos de execução e cada modo é caracterizado por uma duração e uma dada demanda de recursos .

O modelo é amplo o suficiente para permitir manusear o problema que denominamos *problema de otimização de tempo e nivelamento com recursos limitados*. Este problema foi selecionado porque preenche uma lacuna na literatura da administração de projetos: a integração entre as técnicas de nivelamento de recursos e os métodos para a programação de projetos sob restrição de recursos.

O modelo proposto é desenvolvido baseado nas premissas formuladas no item 2.9.2 do capítulo 2. Quando a quantidade disponível de recursos não é suficiente para satisfazer a demanda das atividades concorrentes, torna-se necessário tomar decisões de programação onde as restrições nos recursos são violadas. Estas decisões podem levar ao aumento na duração do projeto além daquela determinada originalmente quando estas restrições são ignoradas. O objetivo é tomar as decisões de programação de forma a minimizar o incremento na duração do projeto, sujeito a restrições de precedências e de recursos.

Por outro lado, o problema do nivelamento de recursos surge quando existem recursos suficientes e toma-se necessário reduzir a flutuação nos padrões de utilização. O objetivo aqui é fazer com que a demanda dos recursos seja o mais uniforme possível.

No problema original de nivelamento, antagonicamente ao problema de alocação de recursos limitados, não existe restrição quanto ao emprego dos mesmos. O processo de nivelamento é realizado através da reprogramação das atividades não críticas dentro dos limites das suas folgas disponíveis, mantendo a duração do projeto fixa.

O conceito do nivelamento de recursos é incorporado ao modelo proposto de alocação de recursos limitados como um mecanismo guia na busca da melhor solução, esperando obter, não tão somente soluções com durações mínimas, como também, de certa forma nivelados.

A seguir são apresentados os vários componentes, bem como a discussão detalhada dos principais conceitos e premissas no qual o paradigma é fundamentado.

3.1 Abordagem Proposta

Como já discutido no capítulo anterior, as técnicas básicas de programação de projetos como o PERT e o CPM, são frequentemente inadequadas quando existem diversas atividades competindo pelos mesmos limitados meios. Nestas condições, a competição pelos recursos resulta em conflitos de programação. O processo que determina qual atividade deve ser programada e qual deve ser preterida é denominado *resolução de conflito de recursos.* A maneira como estes conflitos são resolvidos influenciam a duração total de um projeto, ou diferentes projetos de um conjunto de projetos.

Entretanto, a resolução de conflitos de recursos, a complexidade envolvida na programação de tarefas, e a relativa carência de sucesso com métodos matemáticos de otimização, tem levado frequentemente à busca de técnicas baseadas em heurísticas para a solução deste tipo de problemas.

A premissa básica da maioria dos procedimentos heurísticos para a programação de projetos com restrição de recursos é usar o critério de prioridade para ordenar as atividades, para então, programá-las de modo que as limitações dos recursos não sejam violadas e o menor caminho possível seja obtido.

Neste contexto, um modelo baseado em Algoritmos Genéticos é proposto como uma técnica heurística de solução do problema de programação de projetos multi-modos sob restrição de recursos. Esquematicamente, o sistema para a programação de projetos proposto é da forma representada na Figura 3.1.

O sistema consiste de 4 módulos principais: Interfase com o Usuário, Módulo de Planejamento, Módulo de Programação e Módulo Gerador de Relatórios. Os dois principais elementos do sistema, Módulo de Planejamento e Módulo de Programação, são descritos nos ítens 3.2 e 3.3.

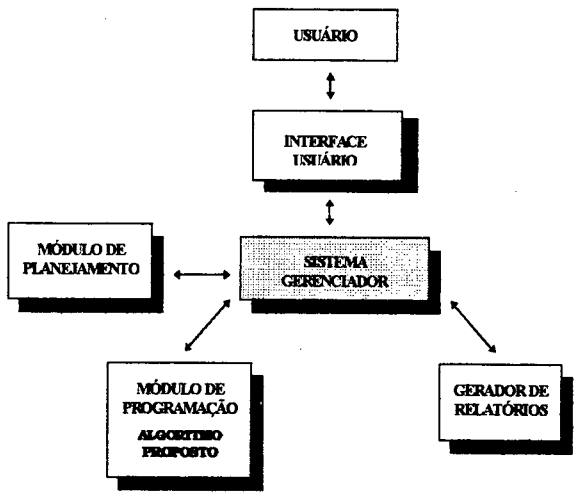


FIGURA 3.1 ARQUITETURA GERAL DO MODELO PROPOSTO

3.2 Módulo De Planejamento

No estágio de entrada o sistema solicita ao usuário, via o módulo Interface com o Usuário, os dados necessários para a aplicação do algoritmo proposto, tais como descrição, relações de precedências, relações duração-recursos (modos) de cada atividade, descrição e níveis de disponibilidade dos diferentes tipos de recursos envolvidos, etc.

A figura 3.2 mostra esquematicamente a arquitetura deste módulo

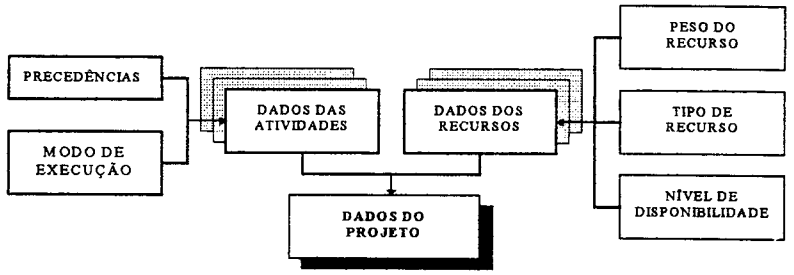


FIGURA 3.2 REPRESENTAÇÃO ESQUEMÁTICA DO MÓDULO DE PLANEJAMENTO

As seções 3.2.1, 3.2.2 e 3.3.3. apresentam uma descrição geral dos diferentes elementos que compõem este módulo e descreve características importantes de cada um deles.

3.2.1 Dados Relativos Às Atividades

Para a execução do sistema proposto é necessário fornecer uma completa identificação das atividades do projeto. As informações solicitadas incluem:

1. *identificação das atividades,*
2. *conjunto das atividades antecessoras de cada atividade,*
3. *relação de dependência entre tarefas precedentes, e*
4. *relação tempo-recursos (modos de execução).*

Os dados requeridos na identificação incluem código e nome da atividade, definição das relações de precedência através do estabelecimento do conjunto de todas as atividades imediatamente antecessoras.

Dentre os quatro tipos de relações de dependência mostrados na Figura 3.3 e que podem ser considerados num diagrama de precedência, somente o tipo tradicional *fim-início* é implementado no modelo (i.e. a sobreposição de atividades não é permitida).

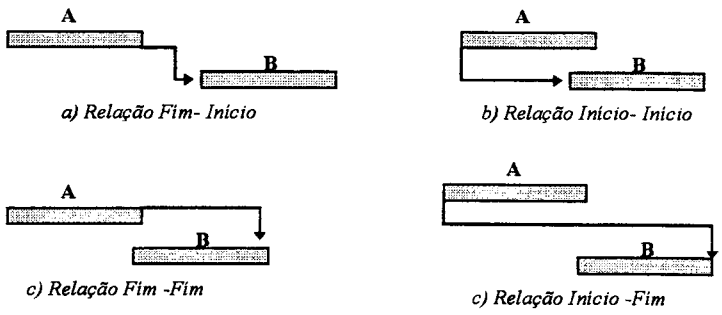


FIGURA 3.3 RELAÇÕES DE DEPENDÊNCIA

Relação Tempo-Recursos (Modos De Execução).

Geralmente, é possível executar uma atividade de várias maneiras, cada uma delas utilizando uma combinação recursos-duração diferente. A despeito da maioria dos modelos tradicionais de programação de projetos com restrição de recursos, onde cada atividade é realizada em, exatamente, um único modo, o modelo desenvolvido incorpora vários modos alternativos para a execução de cada uma das atividades.

A Tabela 3.1. ilustra relações típicas entre duração e demanda de recursos para uma atividade qualquer de um projeto. O exemplo mostra que uma atividade, tanto pode ser realizada em 6 dias por uma equipe de 3, como em 3 dias por uma equipe de 6 tipos de recursos

		Tipos de Recursos				
Modo	**Equipe**	**R1**	**R2**	**R3**	**Duração**	**Custo**
1	3	1	1	1	6	221
2	4	2	1	1	5	232
3	5	2	2	1	4	194
4	6	2	2	2	3	221

Custo diário:
R1 $1.2/hora x 8 horas = 9.6 $
R2 $1.6/hora x 8 horas = 12.8$
R3 $1.8/hora x 8 horas = 14.4$
total 36.8$/dia

TABELA 3.1 EXEMPLO DE RELAÇÕES RECURSOS-DURAÇÕES

No modelo desenvolvido, as relações de recursos-durações para cada atividade, são consideradas como informações estáticas. Informação estática quer dizer que, antes da entrada da relação recurso-duração no sistema, as seguintes premissas foram assumidas:

- A duração de cada atividade, em cada um dos diversos modos de execução, é considerada determinística e permanece constante ao longo da execução da atividade. Uma vez estabelecida a duração, redução ou extensão da mesma não é permitida,
- A quantidade total de trabalho a ser executado permanece constante ao longo da duração da atividade,
- O nível de produtividade de cada modo é constante e conhecido de antemão,
- A combinação de recursos, que leva aos diferentes modos de execução de uma tarefa, é considerado previamente determinado através de qualquer método apropriado,
- Cada tarefa, uma vez iniciada num modo específico, deve ser concluída sem possibilidade de mudança de modo,
- Uma atividade iniciada deve ser totalmente concluída. Isto é, não é permitida a parada temporária de uma atividade em execução, para a liberação dos recursos por ela empregados.

3.2.2 Dados Relativos Aos Recursos

Os recursos são necessários para a execução das atividades e as atividades são os elementos essenciais dos projetos. Portanto, os recursos podem ser considerados como componentes básicos na elaboração da programação de um projeto.

Os atributos relativos aos recursos, requeridos para a implementação do modelo compreendem:

1. *tipos de recursos,*
2. *classes de recursos,*
3. *nível de disponibilidade de cada recurso, e*
4. *importância de cada recurso.*

Tipos De Recursos

Os recursos considerados como entrada do modelo são classificados em quatro categorias: mão-de-obra, material, máquinas e dinheiro. Cada categoria é subdividida em tipos de recursos, como por exemplo, em mão-de-obra pode se ter carpinteiros, pedreiros, serventes, etc.

Dadas as características do trabalho desenvolvido em projetos de construção, frequentemente toma-se necessário o emprego de recursos de diversos tipos, com uma natural importância relativa associada a eles. Por exemplo, alguns recursos podem ser

escassos, outros podem ser caros. Alguns podem necessitar de manuseio especial ou requerer de uma condição particular de operação. Desta forma, os diversos tipos de recursos podem influenciar de forma diferente a progamação de um projeto.

É muito comum que os recursos, sejam eles mão-de-obra, materiais, equipamentos ou dinheiro, tenham uma demanda maior que a oferta, ou sejam limitados por algum critério técnico ou administrativo. Os limites impostos nos recursos essenciais podem afetar significativamente o inicio, execução e término das atividades de uma programação e, muito provavelmente, será necessário estender a duração do projeto além da data desejada para poder atender estes limites.

Entretanto, há situações em que a extensão do projeto além da data de entrega é indesejada. Nestes casos, o problema de programação deve ser tratado como um problema de restrição de tempo, empregando técnicas de compressão de projetos ou de nivelamento de recursos na sua solução. Estas duas últimas situações pressupõem o uso irrestrito de recursos.

No desenvolvimento deste paradigma é proposto o emprego híbrido de técnicas de nivelamento e de otimização da duração de um projeto sob restrição de recursos, como discutido com mais detalhes no ítem 3.3.4.

Classes De Recursos

Por outro lado, na implementação do modelo somente são empregados recursos agrupados na classe de renováveis com disponibilidade constante, seguindo a classificação sugerida por Slowinski (1980), (1981) e Veglarz (1979), (1981), e discutida no ítem 2.7.2. Renováveis são todos aqueles recursos disponíveis em quantidades limitadas, porém que se renovam período a período. Por exemplo, a mão-de-obra: o número de operários disponíveis para trabalhar num projeto é limitada. Porém esta mão-de-obra pode ser renovada a cada período até um patamar pré-determinado.

Nível De Disponibilidade De Recursos

A disponibilidade constante de um recurso está relacionada com o nível de oferta de cada um dos recursos. O nível de disponibilidade refere-se à limitação na quantidade de um determinado tipo de recurso para um dado período de tempo.

Este nível de disponibilidade pode ser chamado de *variável,* se o limite na quantidade de recurso difere de período para período durante a execução de uma determinada atividade. É chamado de *disponibilidade constante,* como nas abordagens clássicas, se a quantidade, para cada tipo de recurso, permanecer inalterada ao longo da vida do projeto.

Importância Relativa De Recursos

Como já colocado linhas acima, dado o uso de diferentes tipos de recursos na execução de uma tarefa, é lógico esperar diferentes níveis de importância entre eles.

Um modelo efetivo para a programação de projetos de construção deve incorporar mecanismos que permitam estabelecer a importância relativa de recursos de natureza diferente. Por exemplo, um modelo deveria ser capaz de distinguir entre recursos limitados pela oferta, recursos limitados por restrições de campo, recursos limitados por restrições técnicas, e recursos limitados pelo custo, atribuindo-lhes pesos diferentes de acordo com a importância de cada um em relação à função objetivo.

Recursos limitados pela oferta são aqueles considerados escassos e difíceis de obter no mercado.

Recursos limitados por restrições de campo são aqueles que o seu emprego durante a execução do projeto está limitado pelas condições de espaço físico. Por exemplo, um projeto desenvolvido numa área grande, pode acomodar mais recursos de um dado tipo do que, outro projeto, confinado a uma área pequena.

Recursos limitados por restrições técnicas são aqueles que o seu uso é influenciado por fatores de natureza técnica ou tecnológica. A modo de ilustração, considere uma atividade que, por uma imposição técnica qualquer, requer o emprego de um guindaste de 10 Tons. e que não pode ser executada usando dois guindastes de 5 Tons. cada um.

Finalmente, um recurso facilmente obtido no mercado, e que por essa razão é considerado ilimitado, pode ser limitado por um critério financeiro. Neste sentido, o recurso pode ser considerado limitado pelo custo.

Por conseguinte, se recursos diferentes são considerados como tendo importâncias diferentes, então é possível assumir que eles podem influenciar uma função objetivo de forma diferente. Recursos escassos podem ter maior influência quando se considera minimização da duração de um projeto, da mesma forma, recursos caros podem ser considerados como tendo maior influência numa função objetivo que trata de minimização de custos.

Outra hipótese viável está relacionada com a possibilidade de usar esta importância relativa dos diversos tipos de recursos, como um meio de comparar diferentes soluções de programação que atingiram o mesmo valor na função objetivo.

É nestas premissas que o modelo desenvolvido se fundamenta para introduzir o nivelamento de recursos, como técnica de ponderação de soluções obtidas através do algoritmo genético.

Estes conceitos e premissas são discutidas na seção 3.3.4

3.3 Módulo De Programação Baseado Em Algoritmos Genéticos

O Módulo de Programação é um modelo protótipo que usa a abordagem de Algoritmos Genéticos para programar as atividades de um projeto de construção. O Módulo de Programação utiliza as informações gerenciadas pelo Módulo de Planejamento, relativas as atividades e recursos, para produzir a denominada *Melhor Programação Possível.*

O desenvolvimento da melhor programação possível é um processo de quatro estágios como ilustrado na Figura 3.4.

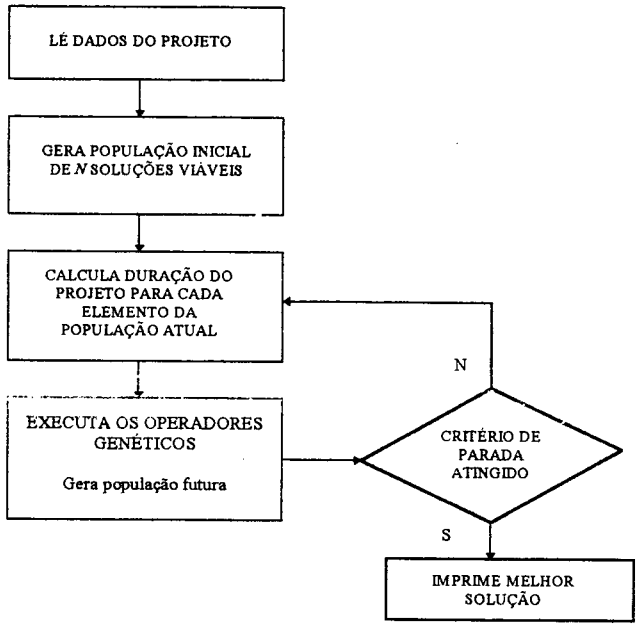


FIGURA 3.4 REPRESENTAÇÃO ESQUEMÁTICA DO MÓDULO DE PROGRAMAÇÃO

No primeiro estágio o módulo de programação *consulta* o módulo de planejamento para a leitura dos dados do projeto.

No segundo estágio é *gerada uma população inicial* de programações viáveis sem se preocupar com a duração final do projeto. Este conjunto de soluções viáveis (população) é gerado, usando uma representação única do problema aqui abordado, pelo denominado *Gerador de Programações Viáveis.*

No terceiro estágio, após se ter o conjunto de soluções viáveis, o módulo de programação *avalia* cada solução com relação à função objetivo, neste caso, a minimização da duração do projeto.

No quarto e último estágio, os operadores tradicionais de *reprodução*, *cruzamento* e *mutação* são aplicados na geração das populações futuras, levando em consideração a flutuação na utilização dos recursos empregados.

Basicamente, o modelo desenvolvido aloca os recursos independentemente para cada solução gerada pelo algoritmo, avalia a performance de cada solução com relação à função objetivo e à utilização dos recursos, e finalmente, recombina as melhores soluções.

A melhor programação possível é obtida após a execução repetitiva (até que o critério de parada tenha sido atingido) dos estágios três e quatro. A melhor solução possível é aquela que apresenta a menor duração com o menor índice de flutuação de utilização dos principais recursos.

Nas seções a seguir, os conceitos acima apresentados são discutidos com maiores detalhes.

3.3.1 Função Objetivo

Para os propósitos deste trabalho, a função objetivo é a *minimização da duração de um projeto de construção* nas bases expostas na seção 2.9.4

3.3.2 Representação Do Problema

De forma a usar a abordagem de Algoritmos Genéticos como heurística de solução do problema em discussão, foi indispensável definir uma representação para o problema única e apropriada, como também foi importante que esta representação permitisse a aplicação *amigável* dos operadores próprios do algoritmo. Por outro lado, especificamente para o problema aqui tratado, procurou-se que a própria estrutura escolhida funcionasse diretamente, como regra de priorização das atividades, a ser empregada na resolução de conflitos de alocação de recursos.

A configuração escolhida, explora a semelhança entre a representação em forma de caracteres concatenados - "string", frequentemente empregados pelas abordagem de Algoritmos Genéticos nos problemas de otimização, e o procedimento de *programação serial* usado no problema de alocação de recursos limitados.

Na abordagem serial para o problema de alocação de recursos, todas as atividades são ordenadas através de uma heurística qualquer, em ordem de prioridade num único grupo. A seguir, as atividades são programadas uma de cada vez, isto é, em série. Por exemplo, no grupo de atividades *CDBA* a atividade *C* é executada primeiro, *D* segundo, seguidas por *B* e *A*.

Na abordagem *paralela,* as prioridades das atividades são determinadas *durante* a execução da programação, ao contrário do que *antes*, como no procedimento serial. Por exemplo, quando uma atividade não pode ser programada num dado período de tempo pela

carência de recursos, ela é transferida para o próximo período onde nova ordem de prioridade para as atividades é estabelecida.

No escopo deste trabalho, a abordagem serial foi considerada mais apropriada à implementação do algoritmo.

De acordo com Falkenauer and Bouffouix (1991), para o problema de programação, o esquema de representação mais apropriado é aquele apresentado na forma de um *string* simples, expressando a sucessão de operações como elas acontecem no espaço de tempo. Este tipo de representação reduz o problema de programação à busca da melhor permutação da operação a ser realizada.

Particularmente, para o problema de programação de projetos, se cada cromossomo é representado como um vetor que apresenta o resultado ao usuário como uma sequência de atividades a serem executadas no qual, a ordem relativa na qual elas aparecem determinam a prioridade de realização, então um projeto de *n-atividades* pode ser representado como:

Início	*Atividade 1*	*Atividade 2*	....	*Atividade n*	*Fim*

Onde os valores, representados em cada uma das celas, são os próprios códigos das atividades, como aparecem no diagrama de precedências do projeto. Na implementação computacional do modelo, por facilidade de manuseio, estes códigos são representados por números inteiros.

É obvio que a estrutura proposta deve representar sequências legais. Para o problema de programação de projetos com restrição de recursos, uma sequência legal ou programação viável, é aquela que satisfaz:

1. a unitaridade, isto é, cada atividade aparece uma única vez no vetor solução, e

2. a integridade das relações de precedência entre tarefas.

Neste aspecto, a programação de projetos é comprovadamente difícil de representar, devido, principalmente, a manutenção da integridade nas relações de precedência.

A figura 3.5 apresenta o diagrama de precedência para um dado projeto. Cabe destacar que a implementação do modelo proposto, exige que todos os projetos a serem tratados possuam uma única atividade de início e uma de fim. Se houver mais de uma atividade inicial ou final, deverá ser adicionada um atividade artificial (*dummy*) no início ou no final do projeto, onde ela for requerida. Atividades artificiais, são atividades que não consomem recursos nem tempo, tendo a função única de auxiliar à construção das relações de precedência.

A figura 3.6 mostra a *solução A* que representa uma programação legal para o projeto ilustrado na figura 3.5 Da mesma forma, uma programação inviável é ilustrada pela *solução B*, onde a relação de precedência é quebrada uma vez, com a atividade 2 sendo programada para ser executada após a sua imediata sucessora 5.

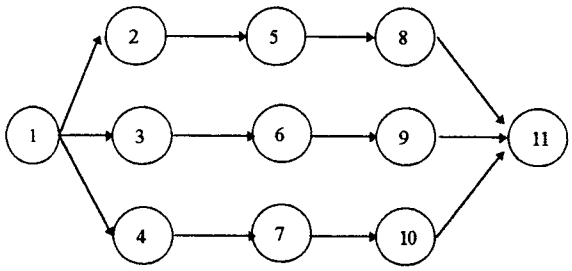


FIGURA 3.5 DIAGRAMA DE PRECEDÊNCIAS PARA UM PROJETO ILUSTRATIVO ADAPTADO DE R.A. JONES (1984)

TABELA 3.2 DADOS PARA O PROJETO ILUSTRADO NA FIGURA 3.5

Atividades	Modos	Recursos Requeridos			Duração	Precedências
		R1	R2	R3		
1		0	5	6	5	-
2		1	3	0	10	1
3		0	3	2	14	1
4		1	4	3	12	1
5		0	5	2	15	2
6		1	4	4	16	3
7		1	5	2	14	4
8		1	5	4	18	5
9		0	5	5	6	6
10		0	5	3	12	7
11		0	5	6	6	8, 9, 10
Disponibilidade de Recursos:		R1 = 1		R2 = 9	R3 = 6	

1	*4*	*7*	*2*	*10*	*3*	*5*	*8*	*6*	*9*	*11*

(A) SEQUÊNCIA DE ATIVIDADES REPRESENTANDO UMA PROGRAMAÇÃO LEGAL

1	*4*	*7*	*5*	*10*	*3*	*2*	*8*	*6*	*9*	*11*

(B) SEQUÊNCIA DE ATIVIDADES REPRESENTANDO UMA PROGRAMAÇÃO ILEGAL

FIGURA 3.6 REPRESENTAÇÕES LEGAIS E ILEGAIS DE UMA PROGRAMAÇÃO

Porém, quando as atividades tem diferentes modos de ser executadas, a representação acima ilustrada não permite acomodar esta premissa. Isto é resolvido, codificando as programações viáveis, como uma lista de atividades contendo sub-listas, representando os diversos modos.

Como cada sub-lista contém os vários modos com que uma atividade pode ser realizada, então, o comprimento de cada uma delas é igual ao número de modos de execução para a atividade considerada.

Por outro lado, já que é pressuposto que cada atividade é executada empregando um, e somente um único modo, logo, cada sub-lista é considerada como sendo uma string formada por *zeros* e um *único um*. A posição na sub-lista representa o modo de execução associado à atividade, enquanto que o valor zero ou um indica se o modo associado ao valor foi atribuído para executar a atividade (valor um) ou não (valor zero). Por exemplo, se uma dada atividade pode ser executada empregando três combinações diferentes de recursos-durações, então, o tamanho da sub-lista é 3. Se o valor *um* aparece na segunda posição isto significa que a atividade será executada empregando o modo dois.

A figura 3.7 ilustra como o problema de programação de projetos para o caso de múltiplos modos é representado na implementação do modelo desenvolvido. A lista representa as atividades de um projeto e a ordem em que a atividade aparece na lista representa a ordem da sua execução, empregando para tanto, o modo representado pelo número um.

Atividade inicial				*Atividade i*				...	*Atividade final*			
0_1	1_2	...	0_m	0_1	...	1_{m-1}	0_m		1_1	0_2	...	0_m

FIGURA 3.7 REPRESENTAÇÃO DO PROBLEMA DE PROGRAMAÇÃO MÚLTIPLOS MODOS

Uma vez que a representação do problema foi definida, faz-se indispensável criar um procedimento, que gere automaticamente o conjunto inicial de programações viáveis, requeridas na execução do Algoritmo Genético. Este é o assunto de estudo da próxima seção.

3.3.3 Geração Da População Inicial De Soluções Viáveis

O modelo implementado, baseado nos conceitos dos Algoritmos Genéticos, torna indispensável definir um único e apropriado mecanismo que permita a geração aleatória da população inicial. Esta população é composta de indivíduos, cada um deles representando uma solução viável ao problema em discussão.

O mecanismo para a geração de cada elemento da população é baseado na metodologia proposta por Jones (1984). Neste paradigma é introduzido uma pequena alteração de forma a acomodar o caso de execução de atividades com múltiplos modos.

A figura 3.8 mostra o diagrama de fluxo para o algoritmo de geração de soluções iniciais utilizados pelo modelo. A descrição de cada um dos passos envolvidos é detalhado através de um exemplo.

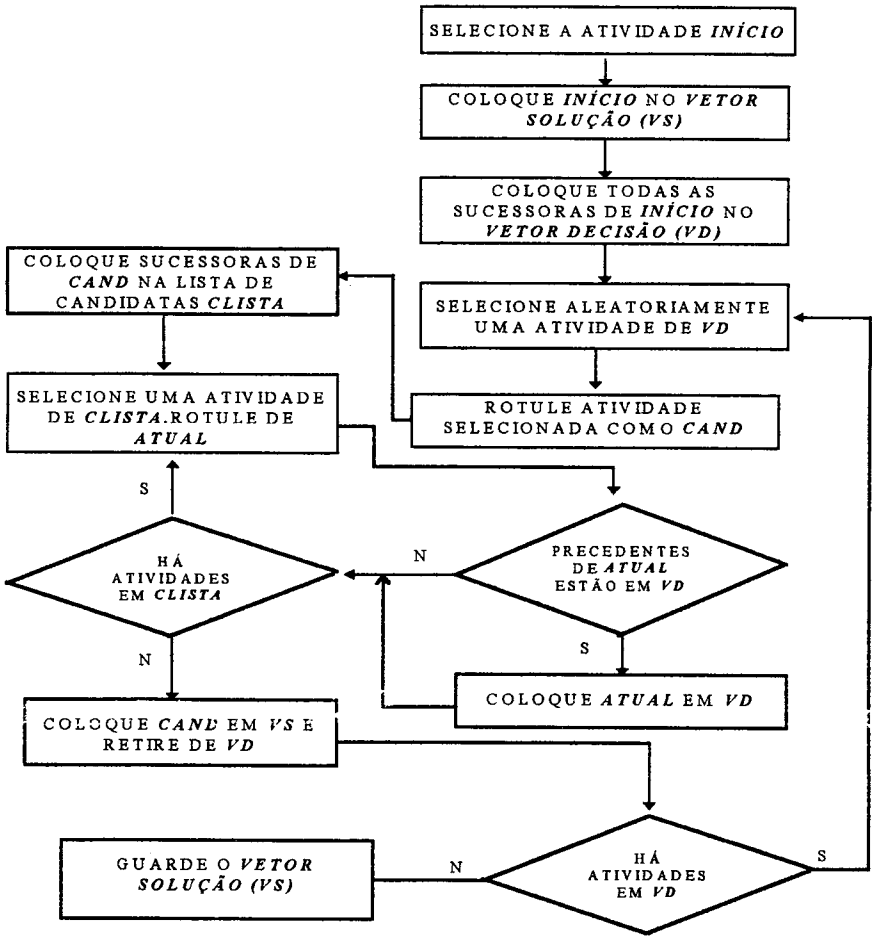


FIGURA 3.8 GERADOR DE SOLUÇÕES INICIAIS VIÁVEIS

Algoritmo Para Gerar Aleatoriamente Um Elemento Da População Inicial

O algoritmo de Jones permite produzir soluções viáveis para o problema de programação de projetos da forma descrita a seguir.

Ao término da execução de uma atividade, a qualquer instante de tempo, é possível definir um conjunto de atividades cujas relações de precedências tenham sidos satisfeitas e, portanto, podem ser consideradas prontas ou elegíveis para a alocação de recursos.

O conjunto de atividades, a partir do qual é feita a escolha da próxima atividade a ser programada, é definido como sendo o *vetor decisão*.

A escolha de uma atividade pertencente ao vetor decisão é realizada aleatoriamente, assumindo-se que o vetor é dividido em celas. O limite superior para cada cela é dada por uma probabilidade, obtida pela divisão da unidade pelo número de atividades elegíveis. Um número fracionário entre 0 e 1, sorteado aleatoriamente, define a cela a ser escolhida, e por

conseguinte, a atividade que será programada. A atividade sorteada é então, incorporada a um vetor denominado *vetor solução,* que representa uma *solução viável* para o projeto em consideração.

A seguir, todas as atividades sucessoras da atividade sorteada, cujas relações de precedência tenham sido cumpridas e que ainda não estejam no vetor decisão, são incorporadas a ele.

O procedimento é repetido sucessivamente até que todas as atividades tenham sido incorporadas, numa sequência arbitrária, ao vetor solução.

A *ordem* de aparecimento das atividades no vetor solução, estabelece uma prioridade implícita para a alocação dos recursos do projeto.

O algoritmo de Jones Modificado é executado de acordo aos seguintes passos:

Passo 1. Selecione a atividade inicial do projeto de construção.

Passo 2. Coloque a atividade inicial no *vetor solução (VS)* e coloque todas as suas atividades sucessoras no *vetor decisão (VD).* O número total de atividades no vetor decisão representa o número de celas em que ele é dividido.

Passo 3. Calcule o limite probabilístico de cada cela, dividindo 1 entre o número de celas e, começando pela primeira cela, adicionando para as celas subsequentes o valor calculado.

Passo 4. Usando um gerador de números aleatórios, sorteie um número entre 0 e 1 e selecione a atividade correspondente no vetor decisão.

Passo 5. Rotule a atividade selecionada aleatoriamente no *passo 4* como CAND, que a identifica como candidata ao vetor solução.

Passo 6. Inclua CAND no Vetor Solução (VS).

Passo 7. Coloque todas as atividades sucessoras de CAND na lista de atividades candidatas CLISTA, de onde a atividade a ser incorporada no vetor decisão (VD) deve ser escolhida.

Passo 8. Selecione uma atividade de CLISTA. Rotule a atividade escolhida como ATUAL.

Passo 9. Verifique se *todas as atividades precedentes* de ATUAL já foram incorporadas ao Vetor Solução (VS). Se verdadeiro, então coloque ATUAL no Vetor Decisão (VD). Uma atividade somente pode ser incorporada ao Vetor Decisão, se todas as suas relações de precedência foram satisfeitas, isto é, se todas as suas atividades antecessoras já foram programadas.

Repita os *passos 8* e *9* sucessivamente até que CLISTA fique vazia.

Passo 10. Para a atividade CAND, escolha aleatoriamente, um e somente um único modo, entre os diferentes modos de execução. Atribua ao modo selecionado o número 1 e faça os modos restantes iguais a zero. No caso da atividade inicial do projeto, o sorteio do modo de execução é efetuado diretamente no *passo 1.*

Passo 11. Retire CAND do Vetor Decisão (VD).

Repetir o procedimento sucessivamente do passo 3 até o passo 10, até que todas as atividades tenham sido selecionadas numa sequência arbitrária. Esta sequência representa uma solução viável para o problema de programação de projetos.

O procedimento é ilustrado usando o projeto cujo diagrama de precedências é mostrado na Figura 3.5 que, por questão de simplificação, é empregado considerando-se somente um único modo de execução.

Inicialmente, selecione a atividade 1 (Inicial) e coloque-a diretamente na Vetor Solução.

VS = [1].

Todas as atividades sucessoras de 1, são incorporadas ao Vetor Decisão, pois as suas relações de precedências são satisfeitas (Atividade 1 já está no Vetor Solução).

VD = [2, 3, 4].

Desta forma, a escolha da próxima atividade a ser incluída no Vetor Decisão será feita entre estas três atividades.

Para a escolha da próxima atividade, os limites de probabilidade de cada cela são determinados. Neste caso, os valores óbvios são: de 0 a 0.33, de 0.33 a 0.67, e de 0.67 a 1.0.

Digamos que o número fracionário (gerado entre 0 e 1) 0.75 foi sorteado. Isto determina que a atividade selecionada é a tarefa 4. A atividade sorteada é retirada de VD e colocada em VS.

VD = [2, 3].

VS = [1, 4].

Todas as atividades sucessoras de 4 são incorporadas na lista de atividades candidatas, CLISTA.

CLISTA = [7].

Particularmente, como CLISTA contém uma única atividade, a atividade de código número 7 é sorteada para ser incorporada ao Vetor Decisão (VD).

Desta forma, a escolha da próxima atividade para o Vetor Solução será feita entre as atividades 2,3 e 7.

VD = [2, 3, 7].

Este mesmo procedimento é repetido até que todas as atividades tenham sido colocadas no Vetor Solução.

Deve ser observado que a escolha de uma atividade do Vetor Decisão não determina a programação atual, mas estabelece a ordem de prioridade para a alocação de recursos para aquela atividade.

3.3.4 Avaliação Das Soluções Geradas

Na abordagem quantitativa do problema da programação de projetos, a avaliação das soluções geradas pelo Algoritmo Genético permite quantificar a aptidão destas soluções. A função objetivo e a função de aptidão ("fitness function") são as formas tradicionais de avaliar a busca da solução ótima e de controlar os operadores genéticos.

Função Objetivo E Função "Fitness"

O problema de programação de projetos é um problema de otimização onde o objetivo final, no caso do modelo proposto, é a redução total da duração do projeto.

A função objetivo típica para este tipo de problema é definida como:

$$\text{Min} \sum_{i=1}^{n} d_i \qquad d_i \in CC \tag{3.1}$$

Onde:

CC = Caminho Crítico

d_i = duração da atividade *i* (*i* = *1, 2, ... , n*).

ou, alternativamente:

$$\text{Min } (x_n - x_1) \tag{3.2}$$

Onde:

x_n = Última Data de Término da atividade final,

x_1 = Primeira Data de Início da atividade inicial.

A avaliação das soluções, geradas inicialmente pelo gerador das soluções da população inicial e, posteriormente, pela aplicação dos operadores do Algoritmo Genético, é simplesmente, a execução do cálculo da Primeira Data de Início (PDI) e Primeira Data de Término (PDT) para cada atividade.

Como já explicado em seções anteriores, a representação proposta para o problema é a heurística empregada na determinação da ordem de prioridades para a alocação dos recursos às atividades. Se a representação proposta para o problema é vista como uma fila

de processamento, então, as atividades são selecionadas da fila de processamento em ordem de prioridade, começando pela primeira tarefa da fila. Quando a fila ficar vazia o processo de alocação termina.

A Primeira Data de Início (PDI) para uma atividade é determinada pela maior Primeira Data de Término (PDT) das suas atividades antecessoras, que dentro do modelo proposto, já foram programadas, tornando elegível a atividade em questão.

$$PDI = PDT_{max} + 1, \quad (3.3)$$

$$PDT = PDI + duração - 1. \quad (3.4)$$

Neste ponto, uma avaliação dos diferentes tipos de recursos solicitados pela atividade corrente, é feita em relação a todas as outras atividades já programadas, e que tenham influência na programação da tarefa em questão.

A Figura 3.9 apresenta a representação esquemática para o processo de alocação dos recursos às atividades e cálculo da duração do projeto

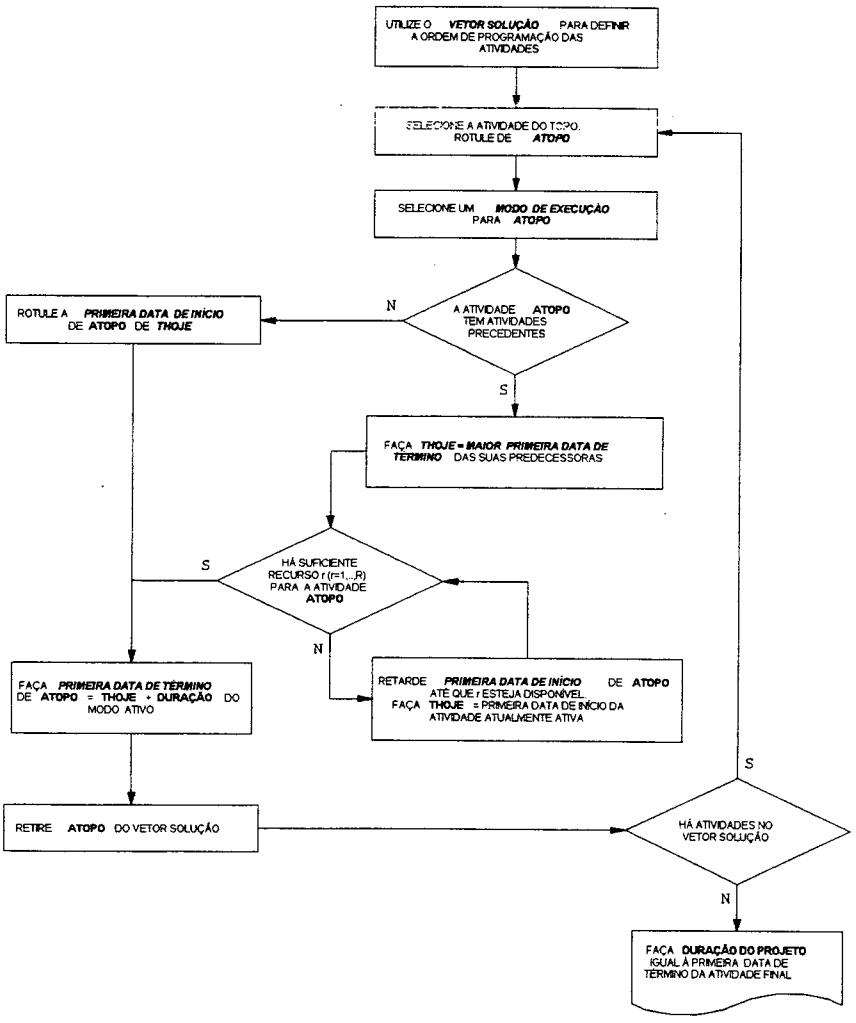


FIGURA 3.9 REPRESENTAÇÃO ESQUEMÁTICA DO PROCESSO DE PROGRAMAÇÃO DAS ATIVIDADES

A Primeira Data de Início de uma atividade, determinada pelas Primeira Datas de Término das atividades antecessoras, será válida somente, se o nível de cada um dos diferentes tipos de recursos empregados pela atividade for suficiente. Se não houverem recursos disponíveis, para qualquer um dos tipos de recursos demandados, a Primeira Data de Início da atividade em questão, será determinada pela primeira data quando o recurso se tornar suficiente.

No caso de alocação de múltiplos recursos, este é um processo iterativo e demorado. Imagine, por exemplo, que hoje um dado tipo de recurso não é crítico e que a atividade necessita, por força de restrição de um outro recurso, ser adiada até uma data onde o nível para o recurso com problemas seja suficiente. Nesta data, o recurso em questão poderá se tornar crítico, solicitando nova e similar avaliação. O processo só termina, quando todos os tipos de recursos envolvidos forem suficientes para atender a demanda de todas as atividades ativas para o intervalo de tempo em consideração.

Ao final deste processo de alocação, a Última Data de Término (UDT), que para a atividade final é igual à Primeira Data de Término (PDT), representa a duração total do projeto. Esta simplificação no cálculo da função objetivo será verdadeira somente, se for convencionado que a Primeira Data de Início da atividade inicial do projeto é o dia 1. Caso contrário, será necessário aplicar a fórmula 3.2 .

A função "fitness" ou função de aptidão ("fitness function)" nos Algoritmos Genéticos é uma medida da qualidade de uma solução para a função objetivo.

Portanto, no caso da função objetivo escolhida para o modelo, a função "fitness" terá uma correlação inversa com a duração do projeto. Isto quer dizer que as programações com durações grandes, terão menor aptidão de reprodução para a próxima geração.

Atualmente, a função de aptidão é um fator de escala, empregado para determinar a qualidade relativa dos elementos (programações) de uma população. A seleção dos elementos dependerá da amplitude da duração esperada e da taxa de mudança nos valores de aptidão.

A função de aptidão escolhida para a implementação do modelo é:

$$y = \begin{cases} C_{\max} - x, x < C_{\max} \\ 0, x \geq C_{\max} \end{cases} \qquad (3.5)$$

Onde, no paradigma aqui desenvolvido, x é a duração calculada para cada elemento (programação) da população atual.

O coeficiente C_{max} é escolhido como sendo o maior valor x da população atual.

Um fenômeno frequentemente observado é que a medida que o algoritmo é executado, a população converge para um conjunto de soluções muito similares ou iguais, entre os quais é difícil discriminar o mais apto se for usada uma função pouco apropriada. Foi observado

que, devido ao grande número de combinações, muitas vezes no problema de programação de projetos, diferentes programações das atividades levam ao mesmo resultado para a função objetivo. A probabilidade de ocorrência desta situação fica mais evidente no caso de múltiplos modos.

Neste contexto, foi introduzido o conceito do nivelamento dos perfis finais dos diferentes tipos de recursos empregados como um procedimento que permite discriminar soluções igualmente aptas, ou soluções muito próximas da aptidão do melhor elemento.

Estes conceitos são discutidos com maior profundidade a seguir.

Medição Da Importância Dos Recursos

Desenvolver um procedimento, que leve em consideração a importância relativa dos diferentes tipos de recursos envolvidos num projeto, pode ser útil como um meio de obter valores mais próximos da realidade quando se mede a eficiência da utilização destes recursos.

Na implementação do modelo proposto, duas distintas aplicações são identificadas para um procedimento deste tipo. Em primeiro lugar, o procedimento é introduzido como um mecanismo que estabelece os diferentes níveis de importância para os perfis de recursos empregados, obtidos após a aplicação da estratégia de programação.

A segunda situação, onde o procedimento proposto é aplicado, está relacionada com a possibilidade de seleção de estratégias que levem a um mesmo valor para a função objetivo. Neste caso, o conceito do nivelamento de recursos é utilizado como regra de desempate. No contexto do paradigma proposto, esta premissa é generalizada para permitir decidir entre estratégias que possuam valores diferentes para a função objetivo.

Este procedimento proposto, associado com a abordagem de Algortimos Genéticos empregada para a otimização da função objetivo, é considerado como um meio útil para reduzir a flutuação no emprego dos recursos do projeto. Em outras palavras, o procedimento possibilita levar em consideração o problema de nivelamento de recursos, ao mesmo tempo que é procurada uma solução para o problema de otimização da duração de um projeto com restrição de recursos.

A situação acima exposta pode ser ilustrada da seguinte maneira:

Inicialmente, considere um projeto que emprega um único recurso R_1 na execução de cada atividade. Suponha que, a aplicação de duas estratégias de programação diferentes E_1 e E_2, levem a um mesmo resultado ($t_1 = t_2$). Suponha que a utilização do recurso para cada uma das estratégias seja representada por uma função f_i (f_1 e f_2).

A Figura 3.10 mostra a situação hipotética onde a estratégia E_1 possui uma utilização mais eficiente do recurso do que a estratégia E_2, i.e. $f_1 > f_2$. Aplicando o conceito de

nivelamento de recursos para esta situação, verifica-se que a estratégia E_1 é melhor do que E_2.

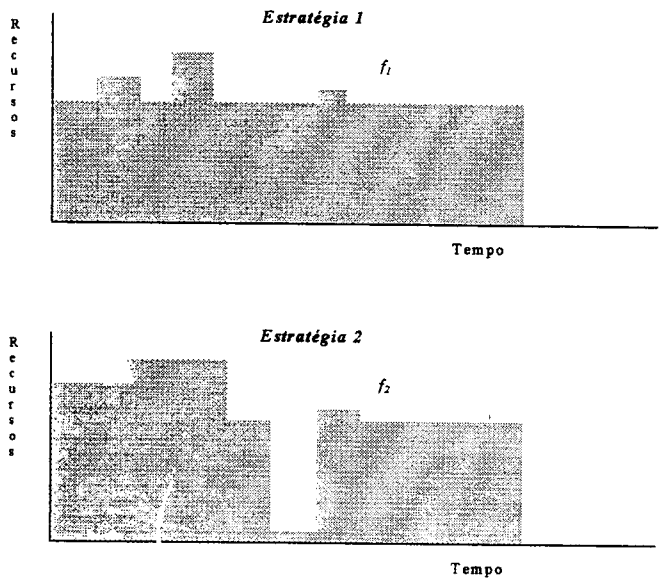


FIGURA 3.10 UTILIZAÇÃO DE RECURSOS EM ALTERNATIVAS DE MESMA DURAÇÃO

Suponha agora que, para este mesmo projeto, dois tipos de recursos sejam considerados. Neste caso, torna-se necessário estabelecer uma relação de importância entre os diferentes tipos de recursos. Suponha que, de uma forma genérica, w_i ($\Sigma w_i = 1$, $i = 1,2,...,n$) represente uma medida do *peso da importância* do recurso do tipo i. No presente exemplo, assuma $w_1 > w_2$.

Suponha que as mesmas duas estratégias, E_1 e E_2, apresentem a mesma duração ($t_1 = t_2$) e gráficos de utilização dos recursos como os mostrados na Figura 3.11. A figura superior ilustra a situação onde, ainda que o gráfico (f_2) do recurso 2 não apresente grande variabilidade, não pode ser considerado como tendo a mesma influência ($w_1 > w_2$) do perfil (f_1) apresentado pelo recurso 1 na seleção desta alternativa (i.e. $w_1 f_1 > w_2 f_2$). A situação oposta é ilustrada na parte inferior da figura.

Se assumir $w_1 = w_2$, então $E_1 = E_2$, levando a escolha de qualquer uma das alternativas. Entretanto se assumir $w_1 > w_2$, isto implica na escolha da estratégia E_2 que apresenta um melhor nivelamento para o recurso mais importante.

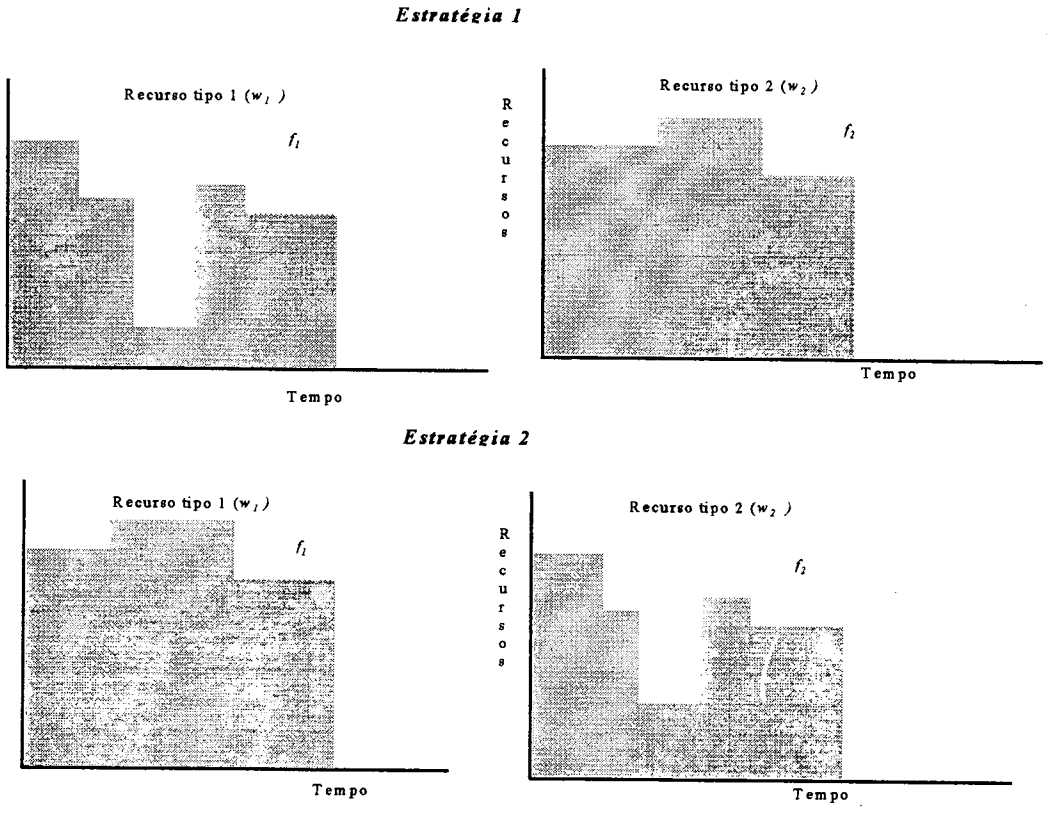


FIGURA 3.11 UTILIZAÇÃO DE MÚLTIPLOS RECURSOS EM ALTERNATIVAS DE MESMA DURAÇÃO

Fator Ponderado De Utilização Dos Recursos

Para a implementação da abordagem proposta é criado o denominado Fator Ponderado de Utilização dos Recursos - FPUR definido como:

$$FPUR_k = \Sigma w_i f_i,\ FPUR_k \in [0,1],\ k = 1,2,...,m \qquad (3.6)$$

Onde:

w_i = peso atribuído para a importância relativa do recurso do tipo i, $\Sigma w_i = 1$;

f_i, = função que mede o índice de nivelamento para o recurso do tipo i, $f_i \in [0,1]$;

k = elemento (programação) da população atual.

A função que mede o índice de nivelamento final de cada recurso empregado pelo projeto é dado por:

$$f_i = \frac{1}{1+V(X)} \qquad (3.7)$$

onde:

$V(X)$ = Variância dos recursos empregados.

Para calcular a Variância, teremos:

$$V(X) = \sum_{t=1}^{T} E[X_t - E(X_t)]^2 \qquad (3.8)$$

onde:

T = Duração do Projeto para a programação atual;

X_t = Soma da necessidade do recurso para todas as atividades do projeto para o período t;

$E(X_t)$ = Soma da necessidade do recurso para todas as atividades do projeto para o período t, dividido pela Duração do Projeto para a programação atual.

A função de nivelamento de recursos, tem como propriedade:

$$\lim_{V(X)\to 0} f_i = 1 \qquad (3.9)$$

$$\lim_{V(X)\to \infty} f_i = 0 \qquad (3.10)$$

As relações 3.9 e 3.10 simplesmente significam que, se um recurso for eficientemente empregado, o padrão final dos recursos empregados terá um comportamento constante, tendendo no limite a ficar totalmente plano. Analogamente, quando o recurso tiver uma utilização ineficiente, ou seja, com uma variância muito grande, este comportamento será refletido por um valor pequeno na função, tendendo para zero no limite.

Este comportamento na função de nivelamento do recurso é refletido, finalmente, no valor fornecido pelo Fator Ponderado de Utilização dos Recursos.

Intuitivamente é evidente, para o caso extremo, que se todos os tipos de recursos são eficientemente empregados, $f_i = 1$, teremos:

$$\text{FPUR} = \Sigma w_i = 1. \qquad (3.11)$$

ou, se todos os recursos são pobremente utilizados, no caso se $f_i = 0$, teremos:

$$\text{FPUR} = 0, \qquad (3.12)$$

Como mostrado no capítulo anterior, a abordagem de Algoritmos Genéticos usa uma probabilidade controlada para aceitar os elementos da população atual que irão gerar a próxima população.

Neste contexto, o modelo implementado propõe o emprego do Fator Ponderado de Utilização dos Recursos (FPRU) como um peso que incrementa ou reduz a probabilidade de aceitação de uma dada solução.

Esta premissa é aprofundada na seção a seguir.

Mudança Na Função De Aptidão

O procedimento acima exposto funciona, indiretamente, como um processo de mudança de escala na aptidão, controlando o número de descendentes, evitando que elementos de alta qualidade se reproduzam excessivamente.

Introduzindo o nivelamento de recursos, a Função de Aptidão (3.5) escolhida para a implementação do modelo, foi modificada para:

$$y = \begin{cases} (C_{\max} - x) - (1 - FPUR), x < C_{\max} \\ 0, x \geq C_{\max} \end{cases} \quad (3.13)$$

Onde FPUR é o Fator Ponderado de Utilização dos Recursos com valores no intervalo [0, 1].

Para ilustrar este conceito, imagine uma população de 5 elementos (soluções) com as características mostradas na tabela 3.3.

Aplicando a Função de Aptidão originalmente proposta (3.5), os elementos possuiriam as aptidões e probabilidades de reprodução mostradas na tabela 3.4.

Entretanto, se o Fator Ponderado de Utilização dos Recursos (FPUR) é introduzido, os valores de aptidão são modificados, levando a uma diferenciação na qualidade das soluções produzidas. No caso da presente ilustração, a probabilidade de reprodução para elementos igualmente aptos desde o ponto de vista da função objetivo, é menor para soluções com uma pobre utilização de recursos, como mostrado na Tabela 3.5 (comparar as soluções 1,2 e 3, 4).

TABELA 3.3 ELEMENTOS DE UMA POPULAÇÃO DE UM ALGORITMO GENÉTICO

Solução No.	Duração	Fator Ponderado de Utilização dos Recursos (FPUR)
1	85	0.99
2	85	0.01
3	86	0.99
4	86	0.01
5	87	0.99
$C_{máx}$ = 87		

TABELA 3.4 APTIDÕES E PROBABILIDADES DE REPRODUÇÃO DOS ELEMENTOS DE UMA POPULAÇÃO

Solução No.	Duração	Fator Ponderado de Utilização dos Recursos (FPUR)	Valor de Aptidão	Probabilidade de Reprodução
1	85	0.99	2	0.333
2	85	0.01	2	0.333
3	86	0.99	1	0.166
4	86	0.01	1	0.166
5	87	0.99	0	0
$C_{máx}$ = 87				

TABELA 3.5 APTIDÕES E PROBABILIDADES DE REPRODUÇÃO MODIFICADAS PELO FPUR.

Solução No.	Duração	Fator Ponderado de Utilização dos Recursos (FPUR)	Valor de Aptidão Modificado	Probabilidade de Reprodução Modificada
1	85	0.99	1.99	0.4975
2	85	0.01	1.01	0.2525
3	86	0.99	0.99	0.2475
4	86	0.01	0.01	0.0025
5	87	0.99	0	0
$C_{máx}$ = 87				

Por outro lado, o emprego do FPUR no cálculo da Função de Aptidão, aumenta a probabilidade de reprodução dos elementos com soluções de qualidade próxima à melhor solução.

O procedimento permite encurtar a distância entre a probabilidade de reprodução de elementos de aptidão inferior, porém, com um uso eficiente dos recursos, dos elementos de aptidão superior e com ineficiente emprego destes recursos.

Por exemplo, no caso das soluções 2 e 3, embora a situação ilustrada na Tabela 3.4 apresente a solução 2 com uma probabilidade bem maior (devido ao valor da função objetivo) do que a solução 3, isto não é considerado como um fator definitivo de decisão se o conceito de nivelamentos de recursos é aplicado. Neste caso, o FPUR é aplicado como um fator de redução da probabilidade de aceitação da solução 2 e de aumento da probabilidade de reprodução da solução 3, como mostrado na Tabela 3.5.

Seguindo esta abordagem, o problema puro de Programação de Projetos com Restrição de Recursos, pode ser considerado como um caso especial (fazendo *FPUR* = 1 em 3.13) do problema de Programação de Projetos com Restrição e Nivelamento de Recursos.

Desta forma, com a introdução deste conceito espera-se obter ao final do procedimento, a sobrevivência de elementos com solução ótima (global ou local) para a função objetivo, e que possuam perfis finais de utilização de recursos de certa forma nivelados.

3.3.5 Operadores Genéticos Do Modelo

Como já salientado anteriormente, do ponto de vista da Pesquisa Operacional, a idéia de um Algoritmo Genético pode ser entendida como a exploração inteligente de uma busca aleatória.

Os Algoritmos Genéticos combinam as noções de evolução e sobrevivência, busca aleatória e estruturada e, geração e avaliação paralela dos nós no espaço de busca.

Como já visto no Capítulo 2, a geração de novos nós de busca de uma geração para outra é realizada usando três operadores básicos clássicos: reprodução, cruzamento e mutação.

Reprodução

A reprodução é um processo de seleção empregado para determinar qual elemento da população atual sobrevirá para gerar novos indivíduos para a próxima geração.

Na implementação do modelo proposto, foi utilizada a técnica de reprodução denominada *elitismo*. O emprego desta técnica assegura que um certo número dos melhores indivíduos da população atual farão parte da nova população.

A função primordial da reprodução elitista é a preservação do melhor material genético presente na população, assegurando muitas vezes a melhoria do desempenho do algoritmo. A segunda função é de servir como um mecanismo de registro ou armazenamento das melhores soluções encontradas ao longo do processo.

A operação de reprodução é uma implementação algarítmica da Teoria de Sobrevivência de Darwin. Através desta teoria sabe-se que a tendência de sobrevivência dos indivíduos mais aptos de uma geração para outra é controlada por alguma regra. Esta é uma regra probabilistica. Ou seja, nem todos os indivíduos mais aptos são automaticamente escolhidos para formar a próxima geração.

No modelo proposto, a geração dos indivíduos da próxima população é feita pelo cruzamento par a par, de elementos da população atual selecionados probabilisticamente.

Para a implementação computacional da seleção dos pais, optou-se pelo processo de seleção conhecido como *roleta*. Neste procedimento, cada setor da roleta representa a aptidão de um individuo da população. A área de uma casa da roleta é proporcional ao valor da aptidão do elemento: quanto maior a aptidão, maior a área que a casa ocupa proporcionalmente às outras casas.

A estrutura básica para a implementação da seleção dos pais usando o mecanismo da roleta é resumida nos seguintes passos:

1. *Some os valores de aptidão de todos os membros da população atual;*
2. *Gere as probabilidades de reprodução de cada individuo da população. Divida a soma calculada no passo 1 pelo valor de aptidão de cada elemento, normalizando desta forma, a área total da roleta.*
3. *Gere **n**, um número aleatório entre 0 e 1;*
4. *Retorne o primeiro membro da população cuja probabilidade de reprodução, somada às probabilidades dos membros precedentes (na população), seja maior ou igual a **n**.*

O mecanismo foi escolhido por ser esta a forma mais simples, eficiente e direta de escolher os pais dos elementos da futura geração, porém, com certeza não é a única maneira, nem a mais rápida.

Cruzamento

O operador de cruzamento pode ser considerado como o operador mais importante na abordagem de algoritmos genéticos. O cruzamento é uma operação complementar da operação de reprodução. A simples cópia de estruturas velhas sem nenhuma mudança, nunca levará a nada novo. É aqui que entra o papel do cruzamento.

O cruzamento é o operador de casamento que permite a produção de novos indivíduos através da troca parcial de informação entre pares de indivíduos. O cruzamento somente acontece com alguma probabilidade P_c , denominada *probabilidade de cruzamento* ou *taxa de cruzamento.*

Como já salientado no Capítulo 2, duas características são essenciais nos algoritmos genéticos no processo de otimização de uma função. A primeira, é a capacidade de convergir para um ótimo (local ou global) após ter localizado a região contendo este ótimo. A segunda característica é a capacidade de explorar novas regiões do espaço de soluções na busca do ótimo global. O equilíbrio entre estas características é ditado pelos valores atribuídos à P_c e à *probabilidade de mutação* P_m , e pelo tipo de cruzamento empregado.

A escolha dos valores de P_c e P_m para o modelo proposto foi baseada nos valores típicos sugeridos na literatura. É bem estabelecido na literatura de algoritmos genéticos que

valores moderadamente grandes para P_c ($0.5 < P_c < 1$), e valores pequenos para P_m ($0.001 < P_m < 0.005$) são essenciais para um desempenho satisfatório do algoritmo genético. Valores moderadamente grandes de P_c promovem a recombinação extensiva dos esquemas. Valores pequenos para P_m são necessários para prevenir o desordenamento de uma solução.

Para o modelo desenvolvido, dentre os operadores clássicos de cruzamento investigados, o *operador CX* ou *operador cíclico* foi o que apresentou melhor desempenho. O operador CX realiza as recombinações no sentido em que cada gene dos descendentes vem da posição correspondente de qualquer um dos pais.

Para ilustrar como o operador CX funciona, usaremos duas soluções (S_1 e S_2) quaisquer para o projeto mostrado da Figura 3.5.

S_1 1 - 2 - 5 - 4 - 7 - 8 - 10 - 3 - 6 - 9 - 11 e

S_2 1 - 3 - 2 - 5 - 8 - 6 - 9 - 4 - 7 - 10 - 11

Neste tipo de cruzamento não ha necessidade de escolher pontos de cruzamento, como os usados no operador de Goldberg - PMX ou no operador de ordem - OX.

Originalmente este procedimento é iniciado da primeira posição mais a esquerda. Na implementação do operador para a representação utilizada no problema em pauta, as duas posições extremas são omitidas, já que representam as datas de início e término do projeto.

Neste sentido, o procedimento é iniciado a partir da segunda posição mais a esquerda, escolhendo um gene (atividade) do primeiro pai (S_1).

S_1' 1 - **2** - ? - ? - ? - ? - ? - ? - ? - ?

Já que por este operador cada gene vem da mesma localização de um dos pais, a escolha da atividade 2 do pai 1 significa que a atividade 3 da oitava posição deve ser escolhida, porque 3 ocupa a segunda posição de S_2 (segundo pai).

S_1' 1 - **2** - ? - ? - ? - ? - ? - **3** - ? - ? - ?

Esta seleção implica que a atividade 4 seja escolhida de S_1,

S_1' 1 - **2** - ? - **4** - ? - ? - ? - **3** - ? - ? - ?

que por sua vez leva a escolher a atividade 5 de S_1, porque 5 está na quarta posição de S_2.

S_1' 1 - **2** - **5** - **4** - ? - ? - ? - **3** - ? - ? - ?

Neste ponto, se continuar com o processo, a escolha da atividade 5 significa ter que selecionar a atividade 2 de S_1. Entretanto, isto não é possível porque 2 já foi selecionado como o segundo gene da solução.

A posição destes genes é dita de formar um ciclo que, eventualmente, retoma ao primeiro gene selecionado. O procedimento para.

Para completar a operação, os espaços vazios são preenchidos com os genes do segundo pai S_2 (mostrados em algarismos itálicos):

S_1' 1 - **2** - **5** - **4** - *8* - *6* - *9* - **3** - *7* - *10* - 11

O segundo filho S_2' é obtido realizando o cruzamento complementar:

S_2' 1 - **3** - **2** - **5** - *7* - *8* - *10* - **4** - *6* - *9* - 11

Observando as soluções produzidas, verifica-se que não são soluções legais, pois relações de precedência foram quebradas em S_1 e S_2:

S_1' 1 - *2* - *5* - *4* - *8* - **6** - *9* - **3** - *7* - *10* - 11

S_2' 1 - *3* - *2* - *5* - **7** - *8* - *10* - **4** - *6* - *9* - 11

No modelo implementado, quando um problema deste tipo ocorre, ele é resolvido *hibridizando* o operador CX, introduzindo a *operação de inversão.*

Esta operação é discutida mais amplamente na seguinte seção.

inversão

O papel desempenhado pela operação de inversão, no caso do paradigma proposto, é de permitir a recomposição das solução ilegais eventualmente produzidas na operação de cruzamento.

A operação de inversão proposta é um procedimento que envolve duas ações distintas: troca de posições e movida, sempre realizadas com pares de genes (atividades).

Inicialmente, o procedimento determina a posição da primeira atividade em conflito. A seguir, uma segunda atividade é selecionada. Esta segunda atividade, ou é a úitima antecessora, ou é a primeira sucessora da atividade considerada, dependendo se a relação de precedência foi quebrada a esquerda ou a direita respectivamente da atividade em questão.

Neste ponto, uma das seguintes ações é executada:

Troca de posições: ocorre quando a posição que ocupa a primeira atividade passa a ser ocupada pela segunda atividade e vice-versa. Para realizar esta operação é necessário que nenhuma relação de precedência da segunda atividade seja quebrada. Isto quer dizer que não pode haver atividades, antecessoras ou sucessoras da atividade em questão, entre a posição atual e a posição destino (posição ocupada pela primeira atividade).

Movida: acontece quando existe algum impedimento de precedência para realizar uma troca de posições. Neste caso, a primeira atividade é deslocada da posição atual para a

posição ocupada pela segunda atividade ou vice-versa. A seguir, todas as atividades localizadas entre as duas posições são movidas uma posição para frente ou para atrás.

O procedimento é repetido até que todas as relações de precedência para todas as atividades sejam recompostas.

No caso das soluções S_1' e S_2', em ambas situações teremos o caso da execução de uma movida para gerar soluções viáveis:

S_1' 1 - 2 - 5 - 4 - 8 - **3** - **6** - 9 - 7 - 10 - 11

S_2' 1 - 3 - 2 - 5 - **4** - **7** - 8 - 10 - 6 - 9 - 11

Mutação

O operador de mutação é empregado com a finalidade de mudar parte da solução para criar uma nova.

O operador de mutação clássico, ou seja, a modificação aleatória pura e simples de um gene, não é aplicável ao problema aqui tratado, devido, principalmente, à destruição das relações de precedência.

No modelo proposto, o operador de mutação implementado segue basicamente o mesmo procedimento usado na inversão, diferenciado por dois aspectos. A primeira diferença reside na probabilidade de mutação P_m que controla a frequência da aplicação do operador.

A segunda diferença está no procedimento empregado para escolher as atividades a serem mudadas. O esquema de perturbação é definido da seguinte maneira:

Dada uma solução viável na população atual, é gerado um número aleatório, *a*, entre 0 e *N*, onde *N* é o número de atividades do projeto. Este número representa a posição da atividade *i* a ser mudada. A seguir, um segundo número aleatório, *b*, é sorteado entre *LI* e *LS*. *LI* representa a posição ocupada pela última antecessora da atividade *i* e, similarmente, *LS* representa a posição da primeira sucessora de *i*, enquanto que *b* designa a nova posição que a atividade *i* passará a ocupar.

Neste ponto, as mesmas duas ações executadas na operação de inversão se aplicam, ou seja, a troca de posições ou a movida.

3.4 Implementação Computacional Do Modelo

O Modelo Evolutivo Para a Programação de Projetos proposto, implementado a nível de protótipo para testes, utiliza a técnica de *programação orientada para objetos - OOP* (OOP = Object-Oriented Programming).

A programação orientada para objetos, tem se mostrado superior em muitos aspectos às abordagens tradicionais usadas para modelar sistemas mais complicados. Uma das mais importantes vantagens da OOP é a de facilitar uma direta e natural correspondência entre o mundo e a sua modelagem. Especificamente na área do gerenciamento de projetos, a filosofia orientada para objetos tem sido comprovadamente de grande ajuda na administração da complexidade dos problemas e tem ganho vários defensores para a representação da engenharia de conhecimento no planejamento e programação da construção (Mosheli, 1993a, 1993b).

A programação orientada para objetos centra a sua base conceitual em quatro principais conceitos: *abstração, encapsulamento, herança* e *polimorfismo.*

A abstração permite delinear as características essenciais das entidades envolvidas na definição de um problema. Estas entidades são denominadas de *objetos*. Na programação orientada para objetos, um objeto é uma entidade abstrata que incorpora as características de um objeto do mundo real. Por exemplo, no presente estudo, distintas características de um projeto, como as atividades e os recursos são representadas como objetos.

O encapsulamento é considerado o conceito central da OOP. O encapsulamento representa a união e o ocultamento de dados ou informações e procedimentos ou métodos dos objetos em estruturas denominadas *classes*. Os dados e os procedimentos descrevem as características e o comportamento de cada objeto.

A herança permite a organização das abstrações em classes de hierarquias nas quais, cada classe tem uma ou mais superclasses imediatas e cada superclasse pode ter várias subclasses imediatas. A herança é provavelmente o conceito que fornece maior poder ao conceito de classes. O emprego desta filosofia fornece grande flexibilidade a baixo custo - em termos de codificação, evitando principalmente a duplicação, limitação ou eliminação e, finalmente, a expansão ou melhora das características da(s) classe(s) ascendente(s).

Por exemplo, no programa desenvolvido, a função objetivo do algoritmo genético é representado por um objeto denominado *CPM*. Este objeto herda características dos objetos *atividades* e *recursos* e adiciona procedimentos próprios que permitem o cálculo da duração do projeto.

Polimorfismo é uma palavra formada por dois termos gregos *poly* (muitos), e *morphos* (formas) significando *múltiplas formas*. O polimorfismo simplesmente descreve a capacidade da codificação da programação orientada para objetos de se comportar diferentemente, dependendo da situação encontrada no tempo de processamento.

O polimorfismo não é uma característica muito associada aos objetos quanto o é aos procedimentos que compõem o objeto. Embora o polimorfismo seja implementado através da arquitetura da classe, somente os procedimentos ou funções membros da classe podem ser polimorfas.

Para ilustrar este conceito usaremos a explanação usada por Faison (1994).

Segundo Faison, este tipo de implementação é similar ao emprego dos verbos na linguagem natural, que é equivalente as funções membros de uma classe. Considere as várias formas que um objeto pode ser empregado na vida real. Considere o verbo *mover*, o qual denota somente uma ação genérica, porque não é sabido qual o objeto que atuará sobre ele. Por exemplo, *mover um lápis* requer ações completamente diferentes do que *mover um "container"*, embora os dois conceitos sejam similares. O verbo *mover* pode ser associado com um conjunto particular de ações somente após conhecer o objeto que atua sobre ele.

Na implementação computacional do modelo, foi utilizado como linguagem de programação o C++ da Borland Inc. (Borland, 1993).

O C++ foi projetado para ser uma melhoria e uma extensão mais poderosa da linguagem C. A extensão mais notável foi a construção de classe. As classes são estruturas que permitem criar tipos definidos pelo usuário, que não somente representam dados, mas também as operações a serem executadas em tais dados (Flamig, 1992). As variáveis criadas pelas classes são chamadas objetos e é o que classifica o C++ como uma *linguagem de programação orientada para objetos.*

As razões para a escolha do C++ são fundamentadas na colocação do Faison. Segundo ele, o C++ representa um excelente balanceamento entre o poder de expressão, velocidade de execução, e tamanho de código. Mais ainda, o C++ da Borland, utiliza o Ambiente de Desenvolvimento Integrado Borland C++ - IDE (IDE = Integrated Development Environment) que permite a conexão automática das quatro fases envolvidas num ciclo de desenvolvimento de um programa: *edição, compilação, "linkagem", e execução,* aumentando a velocidade deste ciclo dramaticamente.

3.5 Planejamento Dos Experimentos

De forma a explorar o desempenho do modelo proposto, o estudo do comportamento do algoritmo foi dividido em duas fases: *fase de testes preliminares* e *fase de testes finais*.

A etapa de testes preliminares oferece a oportunidade de determinar o desempenho relativo dos diversos parâmetros do algoritmo, fase às diferentes características dos problemas considerados. Neste estágio, uma execução piloto do modelo é realizada com a finalidade de ajustar os valores dos diferentes parâmetros em relação aos seus valores pré-estabelecidos.

Na fase de testes finais o comportamento e desempenho do modelo, via busca exaustiva, é avaliado por comparação direta, utilizando problemas disponíveis na literatura para os quais existem resultados registrados da aplicação de outras heurísticas.

3.5.1 Ajuste Dos Valores Dos Parâmetros Do Algoritmo Genético Do Modelo

É bem sabido que os parâmetros que controlam um algoritmo genético podem ter um impacto significativo no seu desempenho e que, muitas vezos, são singulares para cada problema abordado. Os três operadores definidos no algoritmo genético: reprodução, crossover e mutação, determinam os três parâmetros mais importantes para os quais devem ser atribuídos valores. Estes são:

1. *Tamanho da população,*
2. *Probabilidade de cruzamento, e*
3. *Probabilidade de mutação.*

Três outros parâmetros complementares cujos valores devem ser determinados são: *número de gerações, taxa de elitismo* e *probabilidade de mudança de modo.*

A determinação dos valores apropriados, ou ajuste fino, dos parâmetros do algoritmo genético é um processo de tentativa e erro, entediante e sem regras formais definidas para a sua realização.

Este processo de tentativa e erro é, eie mesmo, de natureza combinatorial. O objetivo do presente estudo não é determinar a influência destes parâmetros na busca genética, nem fazer um estudo empírico sobre o assunto e sim, demonstrar a viabilidade de utilizar uma abordagem deste tipo para o problema em pauta. Neste sentido, a determinação dos valores dos parâmetros a serem usados na implementação do modelo limitou-se ao ajuste de valores consagrados na literatura especializada.

Vários estudos, como os de De Jong (1975), Grefenstette (1986), Schaffer *et al.* (1989), Davis (1989), Goldberg (1989), Jog *et al.* (1989), Smith *et al.* (1993), Srinivas e Patnaik 1994), Chan e Tansri (1994) entre outros, determinam que os seguintes valores podem ser apropriados:

1. Tamanho da população: de 30 à 200,
2. Probabilidade de cruzamento: de 0.5 à 1.0,
3. Probabilidade de mutação: de 0.001 à 0.005,

A combinação do número de gerações e o tamanho da população determinam o número máximo de soluções tentadas dentro do espaço total de soluções viáveis. A determinação do tamanho do espaço de soluções para o problema de programação de tarefas com restrição de recursos, por si só, já é um problema que envolve a análise matemática do número de sequências possíveis e de explosão combinatorial.

Para ilustrar este conceito será usado o exemplo, extraído de Elmaghraby e Herroelen (1980) para o caso do problema clássico de *1*-máquina, *n*-tarefas, descrito a seguir.

Seja *(J, R)* uma rede de *N* nós, onde *J* é o conjunto de tarefas e *R* representa as relações de precedência, a qual pode ser particionada em *m* redes paralelas não relacionadas (J_1, R_1), (J_2, R_2), ..., (J_m, R_m). Seja S_i o número de sequências possíveis para a sub rede (J_i, R_i), e que $S_{i,max}$ represente o correspondente número máximo de sequências se $R_i = 0$.

Então o número total de sequências possíveis para a rede *(J, R)* é dada por:

$$S = (N!) / \prod_{i=1}^{m} (S_{i,máx} / S_i) \qquad (3.14)$$

Segundo estes autores, esta relação pode ser usada recursivamente para determinar o número de sequências possíveis em estruturas mais complexas.

Por exemplo, para a rede mostrada na Figura 3.12 representando cinco tarefas e um processador, tem-se S = 15.

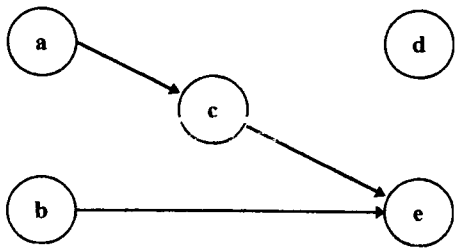
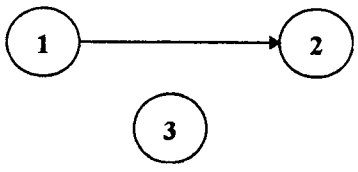


FIGURA 3.12 REDE SIMPLES PARA O PROBLEMA DE 1 PROCESSADOR E 5 TAREFAS. ADAPTADO DE ELMAGHRABY E HERROELEN (1980)

Entretanto, Elmaghraby e Herroelen ressaltam que para o caso da programação de projetos com restrição de recursos, a análise é muito mais complicada e a fórmula (3.14) não é mais totalmente válida.

Por exemplo, para o problema mostrado na figura 3.13 envolvendo 3 atividades e dois recursos, a aplicação de 3.14, leva a *S* = 3. Porém, considerando os dois recursos, o número de sequências possíveis é *S* = 5.



Atividade	Recurso A	Recurso B
1	1 unidade	0 unidade
2	1 unidade	0 unidade
3	0 unidade	1 unidade

FIGURA 3.13 REDE SIMPLES PARA O PROBLEMA DE 3 TAREFAS E 2 RECURSOS. ADAPTADO DE ELMAGHRABY E HERROELEN (1980)

O exemplo tem como objetivo, fundamentar a decisão de descartar a possibilidade de usar uma abordagem algumas vezes empregada na determinação do número total de iterações realizadas pelo algoritmo. A abordagem consiste em fixar o limite máximo para o espaço de soluções a serem testadas, como uma percentagem do espaço total de soluções.

Considerando a variabilidade das características de um projeto de construção em termos do número de atividades, número de tipos de recursos e ainda, número de modos de execução envolvidos, o cálculo de um número máximo de iterações a serem intentadas como percentual do espaço total, acreditamos não ser uma tarefa muito viável.

Por exemplo, se o problema fosse simplificado para o caso de 1 processador e n tarefas independentes, o número de sequências possíveis seria reduzido para o valor máximo de $n!$. Se considerar um problema com 9 tarefas e se for especificado que o espaço de soluções a serem intentadas não exceda 3% do espaço total de soluções viáveis, estes 3% de 9! representam, aproximadamente, 10.000 interações.

Cabe salientar entretanto, que se o espaço de busca for muito pequeno em relação ao tamanho do problema, o algoritmo genético irá convergir muito rapidamente para um sub-ótimo, dado o processamento insuficiente de esquemas. Por outro lado, se o espaço de soluções a serem intentadas é muito grande, o tempo de processamento computacional é demasiadamente grande para se obter uma melhora significante.

Neste sentido, optou-se por escolher uma combinação do número de gerações e do tamanho da população de modo que o número de iterações não exceda 8.000, que representa o emprego de uma população de 200 elementos e um número de gerações de 40. Estes são valores limites bastante observados na literatura especializada.

O número máximo de indivíduos mais aptos transferido para a próxima geração, que representa a taxa de elitismo, foi de 10% do tamanho da população, outro valor tradicionalmente empregado.

A probabilidade utilizada para a escolha dos modos de execução das atividades foi de 100%. Esta probabilidade determina se uma atividade (gene) de um individuo da próxima geração deve ou não mudar de modo. No presente estudo é assumido que sempre os descendentes podem trocar de modo. A escolha do modo de execução de cada atividade é outra seleção feita probabilisticamente. Dado que o número de modos por atividade, nos problemas testados, é pequeno, a probabilidade de escolher o mesmo modo é grande. Este conceito reforça a decisão de utilizar o valor 100% na probabilidade de mudança de modo.

3.5.2 Eficiência Do Modelo

De maneira a gerar os diferentes conjuntos de problemas para a realização do experimento, as características mais importantes dos problemas empregados devem ser

examinadas. Estas características afetam o desempenho dos paradigmas de programação e, portanto, devem fazer parte do planejamento dos testes.

Vários estudos, encontrados na literatura, tratam dos efeitos da estrutura dos problemas no desempenho de modelos para problemas de programação de projetos com limitação de recursos. Entre eles podem ser citados Patterson (1976), Elmaghraby e Herroelen (1980), Kurtulus e Narula (1985), Badiru (1988).

Davis (1973b), classifica as principais medidas das características, para o problema aqui tratado, em três classes gerais:

1. Medidas que caracterizam tamanho, forma e lógica (estrutura de precedências) da rede:

- O *tamanho* da rede é representado pelo número total de nós da rede;
- A *forma* da rede é especificada em termos:
  - do *comprimento* da rede que representa o número máximo de nós consecutivos desde o início até o fim;
  - da *largura* da rede especificada em termos do número máximo de nós em paralelo, e
  - A relação de *comprimento e largura.*
- A *lógica* da rede que representa a *complexidade* da rede;

2. Características de tempo da rede:

- medidas calculadas *antes* da análise do caminho crítico:
  - *soma das durações* de todas as atividades;
  - *duração média* das atividades, e
  - *variância* na duração das atividades.
- medidas calculadas *após* a análise do caminho crítico:
  - *duração do caminho crítico;*
  - *folga total;*
  - *folga livre total;*
  - *densidade* = (soma da duração da atividade) / (soma da duração da atividade + folga livre total).

3. Características relativas aos recursos:

- medidas da demanda dos recursos:
  - *demanda total média por atividade;*

- *demanda média por período;*

- *variância por período;*

- *nível máximo de recursos solicitados;*

- *número de tipos diferentes de recursos.*

• Relação entre demanda e disponibilidade:

- *ociosidade de recursos;*

- *fator de utilização de recursos.*

Maiores detalhes sobre as medidas acima relacionadas podem ser encontradas em Davis (1973).

Além das medidas já relacionadas, outra medida simples que pode ser considerada é o *número de modos* por atividade. De acordo com Patterson *et al.* (1990), a relação que existe entre a duração de uma atividade e o consumo de recursos em projetos multi-modos, faz o problema inerentemente mais difícil de ser resolvido.

De maneira similar, outra medida simples que pode influenciar a complexidade de um problema é certamente, o *número de projetos* executados em paralelo, especificamente quando o caso de múltiplos projetos é considerado.

Das medidas acima descritas, aquelas que representam o tamanho e a complexidade da rede, a duração do caminho crítico sem restrição de recursos, o número de diferentes tipos de recursos e o número de modos por atividade foram empregadas com propósitos comparativos no planejamento do experimento.

Uma das medidas mais importantes e polêmicas é a que representa a complexidade das redes. Parece claro que é necessário medir a complexidade das redes de atividades de forma a estimar os requisitos computacionais e/ou validar procedimentos heurísticos alternativos. Evidentemente, a escolha entre dois algoritmos propostos, ou a determinação da eficiência de um algoritmo particular, será bastante facilitado se existir uma medida (robusta) da complexidade do problema (Elmaghraby e Herroelen, 1980).

A tabela 3.6 apresenta algumas medidas propostas na literatura para a medir a complexidade de redes em problemas de programação de projetos, linha de balanço e programação da produção.

Entretanto, Elmaghraby e Herroelen ressaltam que para o problema de programação de projetos sob restrição de recursos, não se conhece uma medida exata (robusta) que represente a complexidade deste tipo de problema. Segundo eles, parece evidente que a estrutura da rede não é suficiente para refletir a dificuldade encontrada na resolução de tais problemas.

Em particular, a disponibilidade de recursos deve ter um papel importante, e conjectura-se que a relação entre a complexidade da rede e a disponibilidade de recursos deve ser como a ilustrada na Figura 3.14.

Por exemplo, se os recursos estão disponíveis em pequenas quantidades, haverá relativamente pequeno grau de liberdade para programar as atividades, e portanto a complexidade será pequena. Se, por outro lado, os recursos são fartos, as atividades podem ser facilmente programadas em paralelo, levando a uma complexidade igual a zero, para o caso em que a duração obtida seja igual ao comprimento do caminho crítico. O problema é obter a forma exata que a curva da complexidade assume na região entre estes dois pontos extremos. Segundo Elmaghraby e Herroelen, não se conhece nenhuma medida que permita resolver este problema.

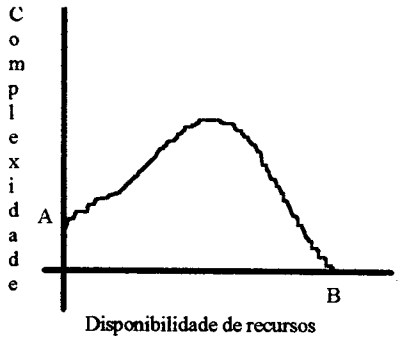


FIGURA 3.14 COMPLEXIDADE DE REDES VERSUS DISPONIBILIDADE DE RECURSOS. ADAPTADA DE ELAMGHRABY E HERROELEN (1980)

Neste contexto, na implementação do modelo proposto, optou-se pela proposta de Badiru (1988), que define a medida de complexidade de um projeto como:

$$\lambda = \frac{p}{d}\left[\left(1 - 1/L\right)\sum_{i=1}^{L} t_i + \sum_{j=1}^{R}\left(\frac{\sum_{i=1}^{L} t_i x_{ij}}{Zj}\right)\right] \qquad (3.15)$$

onde,

λ = complexidade do projeto,

L = número de atividades do projeto,

t_i = duração da atividade i,

R = número de tipos de recursos,

x_{ij} = quantidade do recurso do tipo j solicitado pela atividade i,

Z_j = quantidade máxima disponível para o recurso do tipo j,

p = número máximo de atividades predecessoras na rede,

d = duração do projeto sem restrição de recursos.

TABELA 3.6 MEDIDAS DE COMPLEXIDADE DE REDES
ADAPTADA DE ELAMGHRABY E HERROELEN (1980)

Redes De Atividades	
- Coeficientes De Complexidade De Rede	
CNC(P) = A/N	Pascoe[1]
CNC(D) = $2(A-N+1)/(N-1)(N-2)$	Davies
CNC(K) = A^2 / N	Kaimann
- Densidade Total De Atividade - T-Densidade	
$\sum_{N}$ *máx* { 0; número de atividades predecessoras - número de atividades sucessoras }	Johnson
- Densidade Média De Atividades	
$\frac{T - Densidade}{N}$	Patterson
- Restringência	
$R = 1 - \log S / \log S_{máx}$	Thesen
S = número de sequências possíveis	
Linha De Balanço	
- Sequências Viáveis	
$N! / 2^r$	Ignall
r = número de relações de precedências	
- Robusteza De Ordem	
número de pares ordenados / ($(N (N - 1) / 2)$	Mastor
- Taxa De Flexibilidade	
número de zeros em meia matriz / ($(N (N - 1) / 2)$	Dar-El
Programação De Máquinas	
- "Dependent Shop"	
$MAXPOS(DS) = \prod_{i=1}^{m} \frac{n(j)!}{DEMON(j)}$	Spencer
$DEMON(j) = \prod_{i=1}^{n(j)}$ [1+número de tarefas requerendo máquina j a qual é dominada pelo nó i, onde ,$i,j \in$ SET(j)]	
$j = 1,2, .., m$ máquinas	
SET(j) = nós requerendo a máquina j	
$n(j)$ = total de tarefas requerendo a máquina j	

[1] Reportar-se a Elmaghraby e Herrolen (1980) para as referências dos autores citados.

Badiru explica a expressão proposta da seguinte maneira: O número máximo de predecessores p é um fator multiplicativo que aumenta a complexidade e representa potenciais gargalos na rede. O termo (1 - 1/L) é uma medida fraccionai (entre 0 e 1.0) que indica a intensidade de tempo ou conteúdo de trabalho do projeto. À medida que L aumenta, a quantidade (1 - 1/L) aumenta, e uma fração grande do tempo total requerido (soma de t_i) é adicionado à complexidade da rede. Igualmente, se L decresce, a complexidade da rede decresce proporcionalmente com o tempo total requerido. A soma de (t_{ij}) indica o consumo no tempo de um dado recurso do tipo j relativamente a sua máxima disponibilidade. O termo é somado para todos os tipos de recursos envolvidos. Ter a duração no denominador ajuda a expressar a complexidade como uma quantidade adimensional pelo cancelamento das unidades de tempo do numerador. Além disto, a fórmula expressa a complexidade da rede por unidade da duração total do projeto.

Segundo este autor, qualquer medida para a complexidade de um projeto deveria ser usada como uma medida relativa de comparação mais do que um indicativo absoluto da dificuldade envolvida na programação de um dado projeto.

Para testar uma metodologia, é apropriado comparar o modelo que é proposto contra outras abordagens desenvolvidas para o mesmo problema. Existem hoje, literalmente, centenas de heurísticas diferentes para o problema de programação de projetos sob restrição de recursos e um único modo de execução. Poucas referências são encontradas para o caso de múltiplos modos e, infelizmente, não se tem referências de algoritmos que, explicitamente, reconheça a eficiência da utilização de recursos ao mesmo tempo que a duração do projeto é otimizada. Portanto, o experimento é planejado de forma tal que o caso de um único modo sirva como referencial de comparação do modelo.

Cabe no entanto destacar, que o objetivo final dos testes é demonstrar que a abordagem de algoritmos genéticos pode ser usada combinada com um apropriado mecanismo de nivelamento de recursos, para obter eventualmente (se houver), uma solução ótima, melhorada sob este critério.

Neste sentido, um total de 25 projetos diferentes foram utilizados para testar a eficiência do modelo proposto. Nove projetos foram selecionados de um conjunto de 18 redes, gentilmente fornecidas pelo Professor Raymond Li do Department of Bussiness Systems of the Monash University, Clayton, Austrália, usados no trabalho de Li e Willis (1992). Os outros 16 projetos foram selecionados de um conjunto de 30 redes, fornecidas graciosamente pelo Professor Adedeji B. Badiru, diretor do Expert Systems Laboratory, School of Industrial Engineeering, University of Oklahoma, Norman OK, USA.

Os projetos não utilizados foram rejeitados por empregarem um único recurso para a sua execução, ou por serem muito pequenos (poucas atividades).

Estes projetos foram escolhidos porque, ou a solução ótima é conhecida, ou se tem uma referência do melhor valor fornecido por uma heurística alternativa.

No trabalho de Badiru (1988) o conjunto de 30 projetos foi empregado para comparar o desempenho de 13 heurísticas. Por outro lado, o segundo conjunto de 18 projetos foi empregado no trabalho de Li e Willis onde, o modelo por eles proposto, é confrontado contra 8 procedimentos heurísticos alternativos. Sete destas 8 heurísticas são diferentes das 13 heurísticas testadas por Badiru.

Entretanto, todos estes projetos consideram um único modo de execução para as atividades. Dado que o modelo proposto trata o modo único como um caso especial do problema de múltiplos modos, as redes tiveram que ser adaptadas para poder realizar a análise da efetividade do modelo para este caso mais geral.

As principais adaptações realizadas foram:

- Modos M de execução adicionais para as atividades foram aleatoriamente gerados de acordo com $1 \leq M \leq 3$;
- Nesta fase de implementação do modelo, os pesos para os diferentes tipos de recursos foram diretamente atribuídos pelo decisor ($\sum_{i=1}^{R} w_i = 1$).

Em geral, a intenção de gerar no máximo três modos de execução para cada atividade vêm da idéia de que cada atividade possa ser executada de três formas diferentes: usando o modo de menor duração, o mais provável ou o de maior duração. Por outro lado, os modos foram gerados de tal maneira que os limites dos recursos originalmente impostos aos projetos sejam mantidos inalterados.

O modo de execução original de cada um dos projetos, foi selecionado como sendo o de menor duração. O objetivo está na intenção de realizar uma comparação do resultado obtido pelo modelo para o caso de modo único com o resultado obtido introduzindo múltiplos modos.

Tentar-se-á com isto mostrar que a abordagem proposta pode ser útil também para tratar o problema de compressão de projetos. Este problema tem como premissa básica, similarmente à questão do nivelamento, a não limitação dos recursos empregados. Deverá ser mostrado, na prática, que é possível empregar a abordagem de múltiplos modos para reduzir a duração de um projeto com limitação de recursos.

As abordagens tradicionais para tratar o problema de compressão de projetos fundamentam-se na redução da duração das atividades com o objetivo de reduzir a duração total do projeto. Estes processos usam uma estimativa da variável tempo-custo para cada atividade para determinar qual duração a ser usada de modo a minimizar economicamente a duração do projeto. Estas premissas são válidas quando os recursos são considerados ilimitados.

O problema de compressão de projetos com restrição de recursos para o caso de modo único é tratado em Deckro e Hebert (1989). Entretanto, no presente trabalho é proposto usar

múltiplos modos como abordagem ao problema de compressão. Sob condições de restrição de recursos, a redução da duração de uma atividade não sempre levará a uma redução da duração total do projeto. Obviamente, executar uma atividade com a menor duração viável parece ser a escolha mais apropriada se o desejo é minimizar a duração do projeto. Entretanto, esta regra pode ter também um efeito oposto. Executar uma atividade com menor duração possível requer um alto consumo de recursos, isto pode forçar a retardar atividades que poderiam ser executadas em paralelo. Consequentemente, prolongaria a duração total do projeto.

Neste sentido, para mostrar este conceito, o modelo é executado assumindo que o caso de um único modo representa a menor duração economicamente viável. O resultado obtido nesta execução é comparado com o resultado obtido considerando outras combinações de custo-duração.

Para realizar todos os propósitos acima delineados, foi planejado a execução de dois conjuntos de teste:

1. Conjunto 1: projetos com atividades executadas em uma única forma (modo) predefinida, levando em consideração o nivelamento de recursos e,
2. Conjunto 2: projetos onde cada atividade pode ser executada de um e somente um único modo, selecionado dentre os M modos, considerando a eficiência na utilização dos recursos.

3.5.3 Critérios Para A Avaliação Da Eficiência Do Modelo

O principal critério de avaliação utilizado foi a percentagem média acima ou abaixo do resultado da melhor heurística alternativa reportada.

Um segundo critério usado foi a percentagem média no atraso total do projeto. O atraso de um projeto é a diferença entre o resultado obtido e a duração alcançada na análise do caminho crítico sem restrição de recursos. Esta medida dá uma indicação do atraso introduzido como resultado da limitação na disponibilidade dos recursos e o resultado da heurística empregada. Para o caso de múltiplos modos, o comprimento do caminho crítico é calculado usando o modo de menor duração sem restrição de recursos.

Como terceiro critério, quando a solução ótima para o projeto é conhecida, foi utilizada a percentagem média acima do ótimo.

O critério tempo de processamento computacional foi empregado com limitações. Dado o fato de não ter havido a necessidade de implementar computacionalmente nenhuma heurística alternativa, não foi possível realizar uma comparação direta do tempo de processamento empregado. O tempo obtido pela aplicação do modelo, não pode ser comparado com o tempo reportado nos estudos usados como pontos de referência, dado a

diversidade de fatores envolvidos numa programação (equipamento empregado, eficiência de código, maneira e vícios de programação, linguagem empregada, etc.). As heurísticas teriam que ter sido implementadas no mesmo ambiente em que se desenvolveu a pesquisa para que os tempos computacionais significassem alguma coisa. O principal objetivo deste critério é dar uma idéia do desempenho e limitação do modelo em diferentes condições de teste ou quanto ao tamanho dos problemas tratados.

Outros critérios usados foram: o número de vezes que o algoritmo foi capaz de encontrar a melhor solução, o número de vezes que foi a única heurística a obter a melhor solução, o número de vezes que encontrou a pior solução e o número de vezes que foi a única a obter a pior solução

CAPÍTULO 4

RESULTADOS NUMÉRICOS

Neste capítulo são apresentados os resultados numéricos decorrentes da aplicação da versão computacional do modelo discutido no Capítulo 3.

Este capítulo desenvolve-se dos tópicos básicos, onde são apresentadas as diversas características dos problemas testados e a determinação dos valores empregados nos parâmetros do algoritmo genético, até a apresentação dos resultados obtidos.

4.1 Características Dos Problemas Testados

A fim de testar o desempenho do modelo, 26 diferentes problemas coletados da literatura foram empregados, cujos dados são mostrados no Anexo I. O número de atividades nestes problemas varia entre 7 e 80. Os tipos de recursos empregados variam de 2 a 10 e o número médio de modos de execução por atividade para cada projeto varia entre 1,14 e 2,87. A complexidade das redes varia entre 5,78 e 69,82.

Por outro lado, seguindo a divisão sugerida por Boctor (1990), os projetos foram divididos em dois grupos: problemas pequenos, para projetos com número total de atividades menores de 30 e problemas grandes, para redes com número de atividades maior ou igual a 30. Dentro desta classificação, 7 projetos são considerados grandes e 19 são pequenos.

A Tabela 4.1 resume as características de cada um dos problemas empregados no experimento.

4.2 Parâmetros Do Algoritmo Genético

Como já foi salientado no Capítulo 3, a determinação dos valores apropriados dos parâmetros do algoritmo genético é um processo de tentativa e erro. Na presente aplicação os testes foram efetuados com um único intuito, adequar os valores reconhecidos na literatura aos problemas experimentados.

Neste sentido, a Tabela 4.2 mostra as diferentes combinações do número de gerações e tamanho de população experimentados.

TABELA 4.1 CARACTERÍSTICAS DOS PROBLEMAS TESTE

Problema No.	No. Atividades	Complexidade[1]	Tipos de Recursos	No. médio de modos p/ atividade[2]
1	24	30.91	3	2.41
2	21	12.15	4	2.14
3	15	17.27	4	2.33
4	12	7.33	2	1.75
5	16	8.98	3	2.18
6	40	24.20	5	2.87
7	55	18.13	4	1.87
8	73	69.82	10	1.54
9	27	24.12	3	2.63
10	80	37.24	2	1.43
11	51	35.87	2	1.69
12	7	13.77	4	2.00
13	7	5.85	3	1.71
14	43	14.61	4	1.95
15	13	10.65	4	1.46
16	7	8.10	2	1.62
17	9	13.08	3	2.77
18	12	13.65	3	1.57
19	13	9.62	3	1.14
20	13	7.29	4	1.26
21	16	17.42	2	1.68
22	16	10.8	6	1.47
23	27	19.39	3	1.77
24	30	12.07	6	1.63
25	27	24.12	3	2.77
26	11	5.78	3	2.72

[1]Usando a fórmula proposta por Badiru (1988)
[2]Para o caso de múltiplos modos

Para avaliar a sensibilidade do modelo a estes parâmetros e encontrar a melhor combinação, quatro problemas foram selecionados. Dois projetos pequenos (problemas 25 e 26) com diferentes graus de complexidades e, dois projetos considerados grandes (problemas 6 e 24). Três dos quatro problemas escolhidos possuem solução ótima conhecida. O número de vezes, de um total de 5 execuções por projeto, em que a solução ótima foi atingida, além da análise do incremento da qualidade das soluções obtidas em relação ao tempo computacional empregado, determinou o valor dos parâmetros a serem empregados. Para o caso do problema que não tem solução ótima

conhecida, optou-se pela análise do número de vezes que a menor solução foi obtida. Os resultados são ilustrados na Tabela 4.3.

TABELA 4.2 NÚMERO DE GERAÇÕES E TAMANHO DE POPULAÇÃO EXPERIMENTADAS

	Tamanho Da População	
Número de Gerações	**100**	**200**
20	√	√
30	√	√

TABELA 4.3 CONTAGEM DO NÚMERO DE VEZES QUE O ÓTIMO FOI ATINGIDO

		Pequenos		Grandes	
Geração	**População**	**Problema 25**	**Problema 26**	**Problema 6**	**Problema 24**
20	100	0	5	3	5
20	200	2	5	5	5
30	100	0	5	4	5
30	200	2	5	5	5

Exceto para os problemas 3, 9, 11 e 25 nos quais a combinação geração = 20 e população = 200, apresentou desempenho superior relativo à qualidade das respostas obtidas. Para o resto do experimento foi empregada a combinação 20/100.

Os resultados obtidos, para o total dos experimentos executados, mostraram que o operador de cruzamento CX é mais sensível ao aumento do tamanho da população do que ao incremento no número de gerações. Este comportamento, observado para o teste piloto efetuado com os quatro problemas citados, foi constatado também nas execuções de outros projetos selecionados aleatoriamente no decorrer dos testes. Ficou evidente, através dos resultados observados, que o operador é sensível à qualidade dos elementos da população inicial. Uma das principais desvantagens deste operador, relatadas na literatura, é a convergência muito rápida para soluções (imaturas) sub-ótimas. Entretanto, esta desvantagem não foi evidenciada no decorrer da experiência. Acreditamos que a utilização do Fator Ponderado de Utilização dos Recursos (FPUR), contribuiu para propiciar uma discriminação maior dos elementos das populações, evitando assim, a dominância nas primeiras gerações dos elementos mais aptos.

Os valores empregados para as probabilidades de cruzamento e de mutação, para todos os experimentos, foram P_c = 0.99 e P_m = *0.001*. A taxa de elitismo empregada foi 10% da população.

4.3 Medidas De Desempenho Para O Modelo

Como mencionado no Capítulo 3, a eficiência do modelo é baseada na comparação da melhor duração final observada para o projeto com os resultados publicados nos estudos de Badiru (1988) e Li e Willis (1992).

Cabe ressaltar que o modelo foi aplicado aos 26 projetos utilizados no experimento, para cada um dos dois conjuntos de testes definidos na seção 3.5.2, totalizando 52 problemas. Cada problema foi processado cinco vezes, representado um total de 260 observações.

Além das tradicionais medidas de desempenho relatadas na seção 3.5.3 do Capítulo 3, a seguinte medida também foi empregada:

Taxa de Eficiência Heurística (Badiru 1988). Para cada regra *m*, a taxa para cada problema teste é dada por:

$$\rho_{mn} = \frac{q_n}{PL_{mn}}, \quad m = 1,2,\ldots, M, \qquad n = 1,2,\ldots,N \qquad (4.1)$$

onde,

ρ_{mn} = taxa de eficiência da regra *m* para o problema *n*,

q_n = $\min_m \{PI_{mn}\}$

duração mínima de projeto observada para o problema n,

PL_{mn} = duração do projeto para o problema teste *n* aplicando a regra *m*,

M = número de regras heurísticas consideradas,

N = número de problemas testes.

Se a duração mínima global de um projeto é conhecida, então ela é empregada na expressão ρ_{mn} substituindo o valor de q_n.

A *taxa de melhoria* da duração mínima obtida pelo modelo relativamente à melhor heurística relatada nos estudos referenciados, foi calculada como:

$$\Delta_n = \left(\frac{DH_n - DAG_n}{DH_n}\right)(100) \qquad (4.2)$$

onde,

Δ_n = taxa de melhoria;

DH_n = duração do projeto n para a melhor heurística reportada;

DAG_n = menor duração observada para o projeto n, obtida através do modelo proposto.

Por outro lado, é calculada a *taxa de atraso* de cada projeto, da forma definida na seção 3.5.3 como:

$$A_n = \left(\frac{DAG_n - DCC_n}{DCC_n}\right)(100) \qquad (4.3)$$

onde,

A_n = taxa de atraso;

DCC_n = duração do projeto n calculado sem restrição de recursos;

DAG_n = menor duração observada para o projeto n, obtida através do modelo proposto.

Visando medir a melhoria da solução encontrada pelo modelo proposto (para o caso de modo único) em relação ao emprego de múltiplos modos, foi adotado o critério:

$$SM_n = \left(\frac{DUM_n - DMM_n}{DUM_n}\right)(100) \qquad (4.4)$$

onde,

SM_n = taxa de melhoria com múltiplos modos;

DMM_n = duração do projeto n calculado para o caso de múltiplos modos;

DUM_n = menor duração observada para o projeto n, obtida através do modelo proposto para o modo único.

O uso da soma para cada uma das taxas ρ_{mn}, Δ_n, A_n, e SM_n é uma abordagem prática que permite avaliar o desempenho do modelo sobre o total dos problemas experimentados.

Os valores numéricos destes cálculos são relatados na seguinte seção.

4.4 Resultados Numéricos Da Comparação

Os problemas testados foram divididos em dois grupos, de acordo com a procedência dos resultados a serem comparados: o primeiro grupo envolvendo os problemas testados por Badiru (1988) e o segundo grupo, os problemas experimentados por Li e Willis (1992). As tabelas 4.4 e 4.5 mostram a comparação das durações dos projetos encontradas pelo algoritmo e as durações relacionadas a cada uma das heurísticas estudadas por estes autores.

As tabelas 4.6 e 4.7 ilustram a comparação das taxas de eficiência para os dois grupos de problemas experimentados.

Cabe notar que, inicialmente, todos os problemas foram processados considerando um único modo de execução para as atividades. Em seguida o modelo foi aplicado para o caso de múltiplos modos e calculadas as taxas de melhoria da solução em relação ao único modo. Também foi registrado o incremento, em termos relativos, do tempo computacional empregado para processar cada uma das opções.

A tabela 4.8 ilustra uma compilação das diversas durações encontradas para os projetos. A coluna da melhor heurística representa a duração mínima, reportada nos trabalhos referenciados, para o problema em questão. A duração sem restrição foi determinada usando o método tradicional do Caminho Crítico, relaxando os limites impostos aos recursos. A duração obtida pelo modelo é ilustrada na quinta coluna. Esta duração representa a mínima observada para as cinco iterações realizadas com cada problema, desde que o número de vezes em que ela foi obtida, tenha sido igual ou superior a dois. A tabela é completada com duas colunas que mostram a *duração média de projeto*, observada no processamento do algoritmo, para os casos de único e múltiplos modos.

Baseado nestes dados é calculada a taxa de melhoria da qualidade da solução do problema, obtida por meio do algoritmo proposto, em relação à heurística de desempenho superior para cada problema, como ilustrado na Tabela 4.9.

Por outro lado, a Tabela 4.10 mostra a melhoria das soluções dos problemas, obtidas pela aplicação do modelo em relação a melhor heurística global reportada em cada um dos estudos referenciados (baseado nas conclusões de Badiru e Li e Willis).

ACTIM = Activity-Time, (George Brooks)	MLCT = Minimum Late Completion Time,
ACTRES = Activity-Resource, (Bedworth, 1982)	SPD = Shortest PERT Duration,
TIMRES = Time-Resource, (Bedworth, 1982)	LPD = Longest PERT Duration,
GENRES = *w* ACTIM = (1-*w*) ACTRES, (Whitehouse and Brown, 1979)	HNIS = Highest Number of Immediate Sucessors,
ROT = Resource Over Time, (Elsayed, 1982)	CAF = Composite Allocation Factor, (Badiru, 1984)
MTS = Minimum Total Slack,	AG = Algoritmos Genéticos
MLST= Minimum Late Start Time,	

TABELA 4.4 COMPARAÇÃO DAS DURAÇÕES SOB RESTRIÇÃO DE RECURSOS PARA DIFERENTES REGRAS DE PRIORIDADE - ESTUDO DO BADIRU (1988)

Prob. No.	ACTIM	ACTRES	TIMRES	GENRES 1	GENRES 2	ROT	MTS	MLTS	MLCT	SPD	LPD	HNIS	CAF	AG
1	120	134	125	127	120	126	128	120	129	127	130	127	126	**112**
2	89	88	88	88	88	91	89	89	89	91	88	88	88	**79**
3	365	412	365	412	365	383	416	365	365	383	412	365	383	**367**
4	31	32	33	32	31	32	31	31	33	33	35	32	31	**30**
5	175	173	187	187	187	187	173	175	175	187	187	175	173	**173**
6	76	77	77	77	75	80	79	76	75	84	83	81	78	**71**
7	41	50	46	51	46	37	38	41	38	39	49	38	37	**36**
8	145	168	169	176	153	139	184	145	150	148	170	160	147	**134**
9	80	86	81	86	81	70	86	80	72	75	86	86	70	**67**
10	115	118	115	114	115	113	128	115	112	114	135	121	114	**111**
11	169	174	167	177	169	157	182	169	164	155	197	171	157	**156**
12	17	17	17	17	17	17	17	17	17	17	17	17	17	**17**
13	9	12	9	12	9	12	9	9	9	12	9	12	9	**9**
14	90	93	92	93	87	90	94	90	89	93	90	97	90	**84**
15	24	22	24	24	24	24	24	24	24	24	24	24	22	**20**
16	15	17	17	17	17	17	13	15	15	13	17	15	17	**12**

TABELA 4.5 COMPARAÇÃO DAS DURAÇÕES SOB RESTRIÇÃO DE RECURSOS PARA DIFERENTES REGRAS DE PRIORIDADE - ESTUDO DE LI E WILLIS (1992)

Prob No.	ÓTIMA[1]	Ranking Tecnológico	EST	EFT	LST	LFT	Total Float	ACTIM	ACTRES	Li & Willis	GA
17	23	24	24	25	25	24	24	24	23	24	**23**
18	17	21	24	19	23	21	26	23	18	22	**17**
19	39	47	44	45	39	40	39	39	43	39	**39**
20	13	15	15	16	14	13	14	14	16	13	**13**
21	88	112	112	108	92	100	100	92	92	92	**88**
22	45	50	46	50	46	46	53	46	46	46	**45**
23	35	41	40	47	36	39	47	36	39	38	**35**
24	35	42	39	36	39	36	36	39	35	36	**35**
25	64	70	70	74	73	68	74	73	73	67	**64**

[1] Como reportada na bibliografia.

EST = Ascendente Earliest Start Times	Total Float = Ascendente total Float
EFT = Ascendente Earliest Finish Times	ACTIM = Descendente Activity-Time
LST = Ascendente Latest Start Times	ACTRES = Descendente Activity-Resources
LFT = Ascendente Latest Finish Times	LI and WILLIS = Backward Scheduling

TABELA 4.6 COMPARAÇÃO DA TAXA DE EFICIÊNCIA DE REGRA P_{MN} CONJUNTO DE PROBLEMAS I

Prob. No.	ACTIM	ACTRES	TIMRES	GENRES 1	GENRES 2	ROT	MTS	MLTS	MLCT	SPD	LPD	HNIS	CAF	AG
1	0,933	0,836	0,896	0,882	0,933	0,889	0,875	0,933	0,868	0,882	0,862	0,882	0,889	1
2	0,888	0,898	0,898	0,898	0,898	0,868	0,888	0,888	0,888	0,868	0,898	0,898	0,898	1
3	1	0,886	1	0,886	1	0,953	0,877	1	1	0,953	0,886	1	0,953	0,995
4	0,968	0,938	0,909	0,938	0,968	0,938	0,968	0,968	0,909	0,909	0,857	0,938	0,968	1
5	0,989	1	0,925	0,925	0,925	0,925	1	0,989	0,989	0,925	0,925	0,989	1	1
6	0,934	0,922	0,922	0,922	0,947	0,887	0,899	0,934	0,947	0,845	0,855	0,877	0,910	1
7	0,878	0,720	0,783	0,706	0,783	0,973	0,947	0,878	0,947	0,923	0,735	0,947	0,973	1
8	0,924	0,798	0,793	0,761	0,876	0,964	0,728	0,924	0,893	0,905	0,788	0,838	0,912	1
9	0,838	0,779	0,827	0,779	0,827	0,957	0,779	0,838	0,931	0,893	0,779	0,779	0,957	1
10	0,965	0,941	0,965	0,974	0,965	0,982	0,867	0,965	0,991	0,974	0,822	0,917	0,974	1
11	0,917	0,891	0,928	0,876	0,917	0,987	0,852	0,917	0,945	1	0,787	0,906	0,987	0,981
12	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13	1	0,750	1	0,750	1	0,750	1	1	1	0,750	1	0,750	1	1
14	0,933	0,903	0,913	0,903	0,966	0,933	0,894	0,933	0,944	0,903	0,933	0,309	0,933	1
15	0,833	0,909	0,833	0,833	0,833	0,833	0,833	0,833	0,833	0,833	0,833	0,833	0,909	1
16	0,800	0,706	0,706	0,706	0,706	0,706	0,923	0,800	0,800	0,923	0,706	0,800	0,706	1
Soma	14,80	13,88	14,30	13,74	14,54	14,54	14,33	14,80	14,89	14,49	13,67	13,66	14,97	15,98

TABELA 4.7 COMPARAÇÃO DA TAXA DE EFICIÊNCIA DE REGRA P_{MN} - CONJUNTO DE PROBLEMAS II

Probl. No	Ranking Tecnológico	EST	EFT	LST	LFT	Folga Total	ACTIM	ACTRES	Li & Willis	AG
17	0.958	0.958	0.920	0.920	0.958	0.958	0.958	1	0.958	1
18	0.810	0.708	0.895	0.739	0.810	0.654	0.739	0.944	0.773	1
19	0.830	0.886	0.867	1	0.975	1	1	0.907	1	1
20	0.867	0.867	0.813	0.929	1	0.929	0.929	0.813	0.929	1
21	0.786	0.786	0.815	0.957	0.880	0.880	0.957	0.957	0.957	1
22	0.900	0.978	0.900	0.978	0.978	0.849	0.978	0.978	0.978	1
23	0.854	0.875	0.745	0.972	0.897	0.745	0.972	0.897	0.921	1
24	0.833	0.897	0.972	0.897	0.972	0.972	0.897	1	0.972	1
25	0.914	0.914	0.865	0.877	0.941	0.865	0.877	0.877	0.955	1
Soma	7.75	7.87	7.79	8.27	8.41	7.85	8.31	8.37	8.44	9

TABELA 4.8 COMPARAÇÃO DAS DURAÇÕES PARA OS PROBLEMAS TESTE

Probl. No.	Duração Ótima[1]	Melhor Heurística[2]	Duração Sem Restrição	Melhor Duração Obtida Pelo Modelo Modo Único	Duração Média Modo Único	Melhor Duração Obtida Pelo Modelo Múltiplos Modos	Duração Média Múltiplos Modos
1	-	120	71	112	112	107	107,75
2	-	88	74	79	79	77	78,25
3	-	365	238	367	367	309	312,75
4	-	31	23	30	30	28	28,66
5	-	173	125	173	173	132	133,5
6	-	75	46	71	71	75	76,25
7	-	37	26	36	36	42	42,8
8		139	75	134	136,8	141	142,6
9	-	70	37	67	67	65	67,2
10	-	112	85	111	111	103	103,6
11	-	155	93	158	158,6	140	141
12	-	17	9	17	17	11	11
13	-	9	8	9	9	9	9
14		87	48	84	84	88	88
15	-	22	18	20	20	19	19
16	-	13	11	12	12	12	12
17	23	23	16	23	23	23	23,4
18	17	18	16	17	17	17	17,6
19	39	39	36	39	39	37	37
20	13	13	8	13	13,3	14	14
21	88	92	80	88	88	80	81
22	45	46	28	45	45,2	38	39
23	35	36	33	35	35,2	40	40
24	35	35	33	35	35	37	37,6
25	64	67	31	64	64,6	61	62
26	102	102	60	102	102	86	86,4

[1] Como reportada na bibliografia.
[2] Baseadas nos estudos de Badiru (1988) e Li e Willis (1992).

TABELA 4.9 TAXAS DE MELHORIA DA SOLUÇÃO EM RELAÇÃO À MELHOR HEURÍSTICA

Problema No.	Melhor Heurística Para Cada Problema[1]	Melhor Duração Obtida Pelo Modelo - Modo Único	Taxa De Melhoria %
1	120	112	6.66
2	88	79	10.23
3	365	367	-0.54
4	31	30	3.22
5	173	173	0
6	75	71	5.33
7	37	36	2.70
8	139	134	3.60
9	70	67	4.28
10	112	111	0.90
11	155	158	-1.93
12	17	17	0
13	9	9	0
14	87	84	3.45
15	22	20	9.10
16	13	12	7.69
17	23	23	0
18	18	17	5.55
19	39	39	0
20	13	13	0
21	92	88	4.35
22	46	45	2.17
23	36	35	2.78
24	35	35	0
25	67	64	4.48
Melhoria total (%)			74.02
Melhoria média (%)			2.96

[1] Baseadas nos estudos de Badiru (1988) e Li e Willis (1992).

TABELA 4.10 TAXAS DE MELHORIA DA SOLUÇÃO EM RELAÇÃO À MELHOR HEURÍSTICA GLOBAL

(A) CONJUNTO DE PROBLEMAS I

Problema No.	Melhor Heurística Global[1]	Melhor Duração Obtida Pelo Modelo - Modo Único	Taxa De Melhoria %
1	126	112	11.11
2	88	79	10.22
3	383	367	4.17
4	31	30	3.22
5	173	173	0
6	78	71	8.97
7	37	36	2.70
8	147	134	8.84
9	70	67	4.28
10	114	111	2.63
11	157	158	-0.63
12	17	17	0
13	9	9	0
14	90	84	6.67
15	22	20	9.09
16	17	12	29.41
Melhoria total (%)			100.68
Melhoria média (%)			6.29

[1] Baseadas na heurística proposta por Badiru (1988).

TAXAS DE MELHORIA DA SOLUÇÃO EM RELAÇÃO À MELHOR HEURÍSTICA GLOBAL

(B) CONJUNTO DE PROBLEMAS II

Problema No.	Melhor Heurística Global[1]	Melhor Duração Obtida Pelo Modelo - Modo Único	Taxa De Melhoria %
17	24	23	4.16
18	22	17	22.73
19	39	39	0
20	13	13	0
21	92	88	4.35
22	46	45	2.17
23	38	35	7.89
24	36	35	2.78
25	67	64	4,48
Melhoria total (%)			48,56
Melhoria média (%)			5,39

[1] Baseadas nas heurísticas propostas por Li e Willis (1992).

O desempenho do modelo em relação ao tamanho dos problemas experimentados é ilustrado nas Tabelas 11 e 12.

TABELA 4.11 TAXAS DE MELHORIA DA SOLUÇÃO CONSIDERANDO O TAMANHO DOS PROBLEMAS - CONJUNTO DE PROBLEMAS GRANDES

Problema No.	Melhor Heurística Global[1]	Melhor Duração Obtida Pelo Modelo - Modo Único	Taxa De Melhoria %
6	78	71	8.97
7	37	36	2.70
8	147	134	8.84
10	114	111	2.63
11	157	158	-0.63
14	90	84	6.67
24	36	35	2.78
Melhoria total (%)			31.96
Melhoria média (%)			4.56

[1] Baseadas nas heurísticas propostas por Badiru (1988) e Li e Willis (1992).

TABELA 4.12 TAXAS DE MELHORIA DA SOLUÇÃO CONSIDERANDO O TAMANHO DOS PROBLEMAS - CONJUNTO DE PROBLEMAS PEQUENOS

Problema No.	Melhor Heurística Global[1]	Melhor Duração Obtida Pelo Modelo - Modo Único	Taxa De Melhoria %
1	126	112	11.11
2	88	79	10.22
3	383	367	4.17
4	31	30	3.22
5	173	173	0
9	70	67	4.28
12	17	17	0
13	9	9	0
15	22	20	9.09
16	17	12	29.41
17	24	23	4.16
18	22	17	22.73
19	39	39	0
20	13	13	0
21	92	88	4.35
22	46	45	2.17
23	38	35	7.89
25	67	64	4.48
Melhoria total (%)			117.28
Melhoria média (%)			6.51

[1] Baseadas nas heurísticas propostas por Badiru (1988) e Li e Willis (1992).

A Tabela 4.13 apresenta o desvio das soluções obtidas pelo modelo e pela melhor heurística em relação à duração dos problemas sem a imposição de limites nos recursos (medição da taxa de atraso do projeto).

TABELA 4.13 TAXA DE ATRASO DE PROJETO UTILIZANDO O MODELO E A MELHOR HEURÍSTICA

Problema No.	Duração Sem Considerar Limites Nos Recursos	Melhor Heurística Para Cada Problema[1]	Taxa De Atraso Para A Melhor Heurística (%)	Melhor Duração Obtida Pelo Modelo - Modo Único	Taxa De Atraso Para O Modelo (%)
1	71	120	69.01	112	57.75
2	74	88	18.92	79	6.76
3	238	365	53.36	367	54.20
4	23	31	34.78	30	30.43
5	125	173	38.40	173	38.40
6	46	75	63.04	71	54.35
7	26	37	42.31	36	38.46
8	75	139	85.33	134	78.67
9	37	70	89.19	67	81.08
10	85	112	31.76	111	30.59
11	93	155	66.67	158	69.89
12	9	17	88.89	17	88.89
13	8	9	12.50	9	12.50
14	48	87	81.25	84	75.00
15	18	22	22.22	20	11.11
16	11	13	18.18	12	9.09
17	16	23	43.75	23	43.75
18	16	18	12.50	17	6.25
19	36	39	8.33	39	8.33
20	8	13	62.50	13	62.50
21	80	92	15.00	88	10.00
22	28	46	64.28	45	60.71
23	33	36	9.09	35	6.06
24	33	35	6.06	35	6.06
25	31	67	116.13	64	106.45
Atraso total (%)			1153.45		1047.28
Atraso médio (%)			46.14		41.89

[1] Baseadas nos estudos de Badiru (1988) e Li e Willis (1992).

A eventual melhoria das soluções obtidas para os projetos processados de uma única forma e considerando múltiplos modos, é ilustrada na Tabela 4.14.

TABELA 4.14 MELHORIA DAS SOLUÇÕES OBTIDAS PELO MODELO CONSIDERANDO MÚLTIPLOS MODOS

Problema No.	Melhor Duração Obtida Pelo Modelo Modo Único	Melhor Duração Obtida Pelo Modelo Múltiplos Modos	Taxa De Melhoria Da Solução (%)
1	112	107	4.46
2	79	77	2.59
3	367	309	15.80
4	30	28	6.67
5	173	132	23.70
6	71	75	-5.63
7	36	42	-16.67
8	134	141	-5.22
9	67	66	1.49
10	111	103	7.20
11	158	140	11.39
12	17	11	35.29
13	9	9	0.00
14	84	88	-4.76
15	20	19	5.00
16	12	12	0.00
17	23	23	0.00
18	17	17	0.00
19	39	37	5.13
20	13	14	-7.69
21	88	80	9.09
22	45	38	15.55
23	35	40	-14.28
24	35	37	-5.71
25	64	61	4.69
26	102	86	15.69
Taxa de melhoria (%, somente valores positivos)			163.74
Taxa de não melhoria (%, somente valores negativos)			59.96
Taxa de melhoria total (%, todos)			103.78
Taxa de melhoria média (%)			10.92
Taxa de não melhoria média (%)			5.45
Taxa de melhoria total média (%)			3.99

As Tabelas 4.15 e 4.16 mostram o número de soluções ótimas na geração final, iguais do ponto de vista da função objetivo, porém diferentes em relação ao nivelamento dos recursos, bem como a variação entre o melhor e o pior valor do Fator Ponderado de Utilização de Recursos destas soluções, para os casos de único-modo e multi-modos respectivamente.

TABELA 4. 15 VARIAÇÃO DO FATOR PONDERADO DE UTILIZAÇÃO DE RECURSOS - CASO ÚNICO MODO

Problema No.	Número Médio De Melhores Soluções Na Última Geração	FPUR[1] Melhor	FPUR Pior	Variação Do FPUR (%)
1	5	0.5764	0.5529	4.23
2	3	0.6703	0.6669	0.51
3	5	0.4443	0.4407	0.82
4	1	0.9094	-	-
5	4	0.6207	0.6114	1.52
6	5	0.4443	0.4389	1.23
7	3	0.0931	0.0847	9.92
8	1	0.3255	-	-
9	9	0.3182	0.3086	3.11
10	3	0.3870	0.3672	5.39
11	2	0.7363	0.7237	1.74
12	1	0.7431	-	-
13	1	0.3872	-	-
14	7	0.1902	0.1643	15.76
15	2	0.4766	0.4618	3.20
16	1	0.5783	-	-
17	1	0.3885	-	-
18	1	0.9023	-	-
19	1	0.6899	-	-
20	1	0.8412	-	-
21	2	0.7230	0.6692	9.53
22	2	0.7931	0.7918	0.16
23	5	0.2154	0.1969	9.39
24	6	0.5279	0.5175	1.93
25	1	0.3543	-	-
Variação total das soluções válidas (%)				68.44
Variação média das soluções válidas(%)				4.56

[1] Fator Ponderado de Utilização de Recursos

TABELA 4.16 VARIAÇÃO DO FATOR PONDERADO DE UTILIZAÇÃO DE RECURSOS - CASO MÚLTIPLOS MODOS

Problema No.	Número Médio De Melhores Soluções Na Última Geração	FPUR[1] Melhor	FPUR Pior	Variação Do FPUR (%)
1	2	0.5989	0.5649	6.02
2	1	0.6676	-	-
3	1	0.4844	-	-
4	3	0.9292	0.8872	9.49
5	1	0.6436	-	-
6	2	0.4684	0.4618	1.43
7	1	0.1069	-	-
8	1	0.3399	-	-
9	1	0.2956	-	-
10	2	0.4577	0.4013	14.05
11	1	0.7008	-	-
12	3	0.7462	0.5502	35.62
13	1	0.3872	-	-
14	1	0.2133	-	-
15	8	0.5575	0.4862	14.66
16	3	0.6797	0.5558	22.23
17	1	0.3894	-	-
18	1	0.8647	-	-
19	2	0.7297	0.7215	1.13
20	3	0.8901	0.8495	4.78
21	2	0.5960	0.5387	10.64
22	1	0.8263	-	-
23	2	0.1699	0.1586	7.12
24	1	0.4898	-	-
25	1	0.4298	-	-
Variação total das soluções válidas (%)				127.17
Variação média das soluções válidas(%)				11.56

[1] Fator Ponderado de Utilização de Recursos

O tempo computacional empregado para a solução de cada um dos problemas experimentados está relacionado na Tabela 4.17. Os tempos médios de CPU (486/66mhz) registrados correspondem à combinação de parâmetros do algoritmo genético que apresentou melhor desempenho para a solução de cada um dos projetos.

Entretanto, cabe salientar que a intenção de usar o tempo computacional como critério de desempenho não é o de comparar diferentes heurísticas. No presente trabalho, os objetivos principais de medir o tempo empregado foram: avaliar o desempenho do modelo quanto às características dos projetos experimentados e comparar as soluções usando modo único versus o emprego de múltiplos modos.

TABELA 4.17 TEMPOS DE CPU PARA OS PROBLEMAS EXPERIMENTADOS

Problema No.	Número de Atividades	Complexidade	Tempo Computacional Médio - Único Modo (segundos)	Tempo Computacional Médio - Múltiplos Modos (segundos)
1	24	30.91	267	282
2	21	12.15	207	234
3	15	17.27	351	362
4	12	7.33	66	124
5	16	8.98	204	211
6	40	24.20	413	462
7	55	18.13	572	615
8	73	69.82	1849	1968
9	27	24.12	237	256
10	80	37.24	1379	1478
11	51	35.87	1306	1334
12	7	13.77	106	114
13	7	5.85	16	108
14	43	14.61	465	511
15	13	10.65	133	137
16	7	8.10	44	108
17	9	13.08	46	123
18	12	13.65	111	131
19	13	9.62	141	146
20	13	7.29	130	135
21	16	17.42	169	175
22	16	10.8	173	179
23	27	19.39	223	238
24	30	12.07	266	283
25	27	24.12	470	523
26	11	5.78	117	157
Tempo total (em segundos)			9461	10394
Tempo médio (em segundos)			363.88	399.77
Incremento no tempo computacional médio (%)				9.86

Finalmente, a Tabela 4.18 ilustra o tempo computacional médio discriminado por tamanho e complexidade dos problemas testados.

TABELA 4.18 TEMPOS DE CPU SEGUNDO AS CARACTERÍSTICAS DOS PROBLEMAS EXPERIMENTADOS - ÚNICO MODO

Complexidade	Tamanho Dos Problemas	
	Pequenos (< 30 atividades)	Grandes (≥ 30 atividades)
< 18.16*	134.27[1]	434.33
≥ 18.16*	299.25	1236.75

*Complexidade média
[1]Tempo em segundos

CAPÍTULO 5

CONCLUSÕES E RECOMENDAÇÕES

5.1 Conclusões

Este trabalho demonstra que os algoritmos genéticos fornecem uma excelente metodologia para resolver o problema de programação de projetos de construção. O modelo desenvolvido é formulado através de uma representação adequada das soluções do problema e dos operadores do algoritmo, explorando a formulação e a função objetivo para estimar a qualidade das soluções.

O paradigma proposto aborda o problema de alocação de recursos limitados, considerando que cada atividade pode ser executada de uma única maneira, a qual é escolhida entre várias alternativas. A implementação do modelo considera o problema tradicional, de programação de projetos com restrição de recursos e modo único de execução, como um caso especial do problema de múltiplos modos.

Existem atualmente, várias boas heurísticas convencionais para resolver o problema mais tradicional de programação de projetos. Para demonstrar a competitividade da abordagem proposta, estudos de comparação foram realizados. Estes estudos foram baseados na qualidade das soluções obtidas para cada um dos 25 problemas de modo-único experimentados, comparando os resultados alcançados com os resultados publicados por Badiru (1988) e por Li e Willis (1992) utilizando os mesmos problemas.

Os experimentos mostraram que a abordagem desenvolvida representa uma boa alternativa relativa a outras técnicas heurísticas empregadas na solução do problema em questão. Os resultados obtidos demonstram que o modelo fornece soluções robustas e de melhor qualidade do que a melhor heurística reportada, ao menos para a função objetivo considerada (minimização da duração do projeto).

Os resultados mostram que o modelo supera em média a heurística com melhor desempenho global (sobre todos os projetos), em 6.29% para os problemas testados por Badiru e em 5.39% para os problemas experimentados por Li e Willis. O modelo apresenta um desempenho 6.51% superior à melhor heurística nos problemas considerados pequenos e 4.56% nos problemas grandes.

A taxa de eficiência do algoritmo, calculada para o conjunto de problemas testados por Badiru foi de 15.98 sobre um total de 16. Este valor é superior aos 14.97 encontrados para a regra proposta por este autor. Para o conjunto de problemas experimentados por Li e Willis, a taxa de eficiência encontrada foi de 9 sobre 9, relativamente aos 8.44 da heurística de melhor desempenho.

Outra medida de desempenho testada foi a taxa de atraso do projeto. Outra vez o algoritmo proposto apresentou melhor desempenho, obtendo um atraso médio para o total de problemas testados de 41.89% contra os 46.14% das melhores heurísticas individuais, ou seja, somando o atraso das heurísticas que obtiveram o menor resultado individual para cada projeto.

É demonstrado que o modelo desenvolvido, resolve a questão de programação de projetos com recursos limitados e paralelamente, aborda o problema de nivelamento dos mesmos, embora ambas questões, como mostrado anteriormente, sejam consideradas antagônicas. O algoritmo proposto consegue mostrar a viabilidade de tratar os dois problemas simultaneamente.

Uma das principais vantagens de utilizar a abordagem de algoritmos genéticos, no tratamento do problema de otimização de tempo com restrição e nivelamento de recursos, é a capacidade que a metodologia oferece de explorar diversas soluções em paralelo. Esta característica permite obter soluções ótimas, desde o ponto de vista da minimização da duração do projeto, quanto niveladas em relação à utilização dos recursos empregados.

Além dessa vantagem, a implementação do modelo mostra que o conceito de nivelamento de recursos pode ser utilizado como mecanismo de decisão, discriminando soluções igualmente boas (se existirem) quanto ao objetivo principal. Por outro lado, este conceito possibilita introduzir uma maior flexibilidade ao processo decisório, permitindo optar por soluções próximas do ótimo, porém de melhor desempenho quanto a eficiência da utilização dos meios.

Uma constatação observada na execução dos experimentos, que pode ser interpretada como mais uma vantagem decorrente da exploração em paralelo de soluções, foi a viabilidade de obter melhoria na utilização dos recursos através do próprio ordenamento das atividades. Os métodos tradicionais empregados na solução do problema de nivelamento de recursos são míopes neste sentido, pois fundamentam-se exclusivamente na exploração das folgas das atividades, isto é, atrasando atividades não críticas dentro dos limites determinados pelas suas folgas.

Este conceito pode ser ilustrado através do resultado obtido para o problema 26, um dos mais simples do conjunto de projetos experimentados, mostrado na Figura 5.1a e 5.1b.

Este exemplo mostra que, mesmo existindo limitação nos recursos empregados, é perfeitamente possível obter uma melhor solução nivelada, unicamente olhando para a ordem de execução das atividades (observar a ordem de realização das tarefas 8 e 9).

Embora as duas soluções encontradas para o exemplo tenham apresentado uma diferença muito pequena no fator de utilização de recursos, ele é útil no sentido de mostrar a validade do critério empregado.

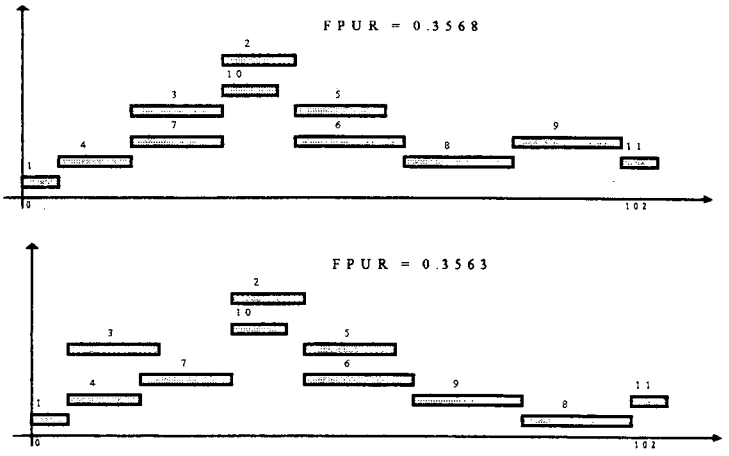


FIGURA 5.1 PROGRAMAÇÕES ÓTIMAS PARA O PROBLEMA 26

No caso dos problemas experimentados, os resultados das tabelas 4.15 e 4.16 do capítulo anterior mostram que se for considerado o nivelamento dos recursos como critério de desempate de soluções iguais, pode-se obter uma melhoria da ordem dos 4.5 % para o problema de programação com modo único e de 11.5 % para o múltiplos modos. Estes valores foram obtidos assumindo que todos os recursos são igualmente importantes.

Finalmente, os resultados alcançados pelo modelo para o caso de programação de múltiplos modos, mostram a viabilidade da sua utilização como um meio que, eventualmente, obtém soluções melhores do que se for considerado o caso de modo único. Não existe, ao menos na literatura consultada, referência que mostre a vantagem prática de usar múltiplos modos ao invés de único modo. Tradicionalmente, os poucos modelos desenvolvidos para tratar múltiplos modos limitam-se a comparar a desempenho deles contra algoritmos alternativos.

O trabalho mostra, dentro das premissas de desenvolvimento dos experimentos, que em 58% dos casos foi obtida uma melhoria da solução único-modo da ordem dos 11% e que, em 42% dos problemas não houve melhoria (15% não sofreram alteração e 27% tiveram piora) com uma taxa média de não melhoria da ordem dos 5%. Estes resultados permitem concluir que a consideração de vários modos de execução para as atividades, realmente fornece maior flexibilidade ao processo de alocação de recursos limitados, levando inclusive, a uma eventual melhora das soluções obtidas para o problema de modo único.

5.2 Recomendações

O algoritmo foi desenvolvido visando atingir os objetivos estabelecidos nas etapas preliminares. Entretanto, o modelo tem imperfeições que devem ser corrigidas para torná-lo mais prático e eficiente.

Trabalhos futuros deveriam ser focalizados na redução do tempo que o algoritmo leva para realizar a busca. O modelo produz soluções superiores aos métodos alternativos, como

mostrado anteriormente, porém o custo é um tempo computacional relativamente longo. As investigações devem passar pela auferição de valores mais apropriados para os parâmetros do algoritmo, bem como a experimentação de novos operadores. Dado que na área da construção cada projeto representa um problema particular, recomenda-se fortemente a investigação de parâmetros denominados adaptativos.

Uma segunda área de pesquisa é a experimentação de outras funções objetivos. Funções alternativas, embora não abordadas no estudo, podem facilmente ser incorporadas ao modelo através de simples modificações do código computacional implementado. Funções tais como *minimização do máximo atraso* $L = \max_i [t_i - d_i]$ (t_i é a duração do projeto e d_i é a data de entrega do projeto), ou a *minimização do custo total do projeto,* seriam particularmente fáceis de serem adaptadas. Prováveis alterações a serem introduzidas no código computacional atual, para uma programação com mínimo custo seriam: substituição da atual função objetivo de minimização da duração, por uma função objetivo apropriada de valores monetários, e mudança das durações das atividades pelos seus respectivos custos totais (custos diretos + custos indiretos).

Recomenda-se aprofundar as investigações sobre a viabilidade de utilizar a abordagem de algoritmos genéticos e o problema de programação com múltiplos modos, como uma alternativa para tratar o problema de compressão de projetos sujeitos à restrição de recursos. Tipicamente, os modelos que tratam desta questão, não consideram a limitação de recursos. Por outro lado, dos poucos trabalhos publicados sobre este tema que incluem limitação de recursos, nenhum aborda o tema como programação de múltiplos modos, embora conceitualmente o problema seja uma questão de seleção de alternativas econômicas, mutuamente exclusivas, para a realização das atividades.

É também conveniente que seja validada a eficiência do modelo na programação de múltiplos modos, através da comparação com algum dos poucos algoritmos especificamente desenvolvidos nesta área ou, em seu defeito, recomenda-se a sua confrontação com a melhor heurística, dentre as pesquisadas por Boctor (1993).

Um outro campo de aplicação do modelo que pode ser investigado é a programação de múltiplos projetos com limitação de recursos. Este problema é considerado uma extensão da programação de um único projeto, porém muito mais complexo porque as atividades pertencem a vários projetos simultaneamente realizados. Recomenda-se, inicialmente abordar os vários projetos como um único grande projeto, deste modo o modelo pode ser aplicado com poucas alterações.

Finalmente, embora o domínio do problema desta pesquisa tenha sido a alocação de recursos para projetos de construção, os princípios e bases teóricas desenvolvidas não estão limitadas a esta área específica. Realmente, eles podem ser aplicados a outras áreas da engenharia e a administração onde o domínio do problema tenha características similares.

BIBLIOGRAFIA

ADAMS, J., BALAS, E. e ZAWACK, D. The Shifting Bottleneck Procedure For Job Shop Scheduling. *Management Science*. v.34, n.3, p. 391-401. 1988.

ADELI, H. An Overview Of Expert Systems In Civil Engineering. In: Expert Systems In Construction And Structural Engineering. Adeli, H. Chapman and Hall, New York. p. 45-84. 1988.

ADIRI, I., FROSTIG, E. e RINNOOY KAN, A.H.G. Scheduling On A Single Machine With A Single Breakdown To Minimize Stochastically The Number Of Tardy Jobs. *Naval Research Logistics*, v. 38, p. 261-271. 1991.

ALIDAEE, B. A Heuristic Solution Procedure To Minimize Makespan On A Single Machine With Non-Linear Cost Functions. *Journal of Operational Research Society*, v. 41, n.11, p. 1065-1068. 1990.

ALIDAEE, B. e AHMADIAN, A. Two Parallel Machine Sequencing Problems Involving Controllable Job Processing Times. *European Journal of Operational Research*, v. 70, p. 335-341. 1993.

APPLEGATE, D. e COOK, W. A Computational Study Of The Job-Shop Scheduling Problem. *ORSA Journal of Computation*. v. 3, p. 149-156. 1990.

ARKIN, E.M. e ROUNDY, R.O. Weighted-Tardiness Scheduling On Parallel Machines With Proportional Weights. *Operations Research*, v. 39, n. 1, p. 64 - 81. 1991.

ASHLEY, D.B e LEVITT, R.E. Expert Systems In Construction: Work In Progress. *Journal Of Computing In Civil Engineering*. v. 1, n. 4, p. 303-311. 1987.

ASSED, J.A. *Construção Civil: Viabilidade, Planejamento, Controle.* Livros Técnicos e Científicos Editora S.A., Rio de Janeiro. 1986.

BADIRU, A.B. Toward The Standardization Of Performance Measures For Project Scheduling Heuristics. *IEEE Transaction On Engineering Management*. v. 35, n. 2, p. 80-89. 1988.

BAKER, K. e SCUDDER, G.D. Sequencing With Earliness And Tardiness Penalties: A Review. *Operations Research*. v. 38, p. 22-36. 1990.

BAKSHI, M.S. e ARORA, S.R. The Sequencing Problem. *Management Science*. v. 16, p. B247-B263. 1969.

BARKER, K.R. *Introduction To Sequencing And Scheduling.* John Wiley, New York. 1974.

BARNES, J.W. e LAGUNA. M. A Tabu Search Experience In Production Scheduling. *Annals of Operations Research*. v. 41, p. 141-156. 1993.

BELL, C.E. e PARK, K. Solving Resource-Constrained Project Scheduling Problems By A* Search. *Naval Research Logistics*. v. 37, p. 61-84. 1990.

BELL, C. E. e HAN, J. A New Heuristic Solution Method In Resource-Constrained Project Scheduling. *Naval Research Logistics*. v. 38, p. 315-331. 1991.

BELLMAN, R., ESOGBUE, A.O. e NABESHIMA, I. *Mathematical Aspects Of Scheduling And Applications*. Pergamon Press. 1982.

BENNINGTON, G.E. e McGININIS, L.F. A Critique Of Project Planning With Resource Constrain, In: *Symposium on the theory of scheduling and its applications*. ed. Elmaghraby, S.E. Lectures Notes in economic and mathematical systems. Springer-Verlag. p. 1-38. 1973.

BIEGEL, J.E. e DAVERN, J.J. Genetic Algorithms And The Job Shop Scheduling. *Computers and Industrial Engineering*. v. 19, n. 1 e 4, p. 81-91. 1990.

BLANCHARD, D. Conference Report WCCI'94: It's A Fuzzy World Out There. *Expert Systems*, v. 11, n. 3, p. 179-180. 1994.

BLAZEWICZ, J., LENSTRA, J.K. e RINNOOY KAN, A.G.H. Scheduling Subject To Resource Constraints: Classification And Complexity. *Discrete Applied Mathematics*. v. 5, p. 11-24. 1983.

BLOCHER, J.D. e CHAND, S. Scheduling Of Parallel Processors: A Posterior Bound On LPT Sequencing And A Two-Step Algorithm. *Naval Research Logistics*. v. 38, p. 273-287. 1991.

BOCTOR, F.F. Some Efficient Multi-Heuristic Procedures For Resource-Constrained Project Scheduling. *European Journal of Operational Research*. v. 49, p. 3-13. 1990.

BOCTOR, F.F. Heuristics For Scheduling Projects With Resource Restrictions And Several Resource-Duration Modes. *International Journal of Production Research*, v. 31, n.11, p. 2547-2558. 1993.

BOWMAN, E.H. The Scheduling-Sequencing Problem. *Operations Research*. v. 7, n. 5, p. 621-624. 1959.

BORLAND INTERNATIONAL INC. Borland C++ 4.0 for DOS, Windows, and Windows NT. 1993.

BRUNS, R. Direct Chromosome Representation And Advanced Genetic Operators For Production Scheduling. *Proceeding Of The Fifth International Conference On Genetic Algorithms*. Urbana, IL. 352-359. 1993.

BUXEY, G. Production Scheduling: Practice And Theory. *European Journal of Operational Research*. v. 39, p. 17-31. 1989.

CADAMBI, B.V. e SATHE, Y.S. Two-Machine Flowshop Scheduling To Minimize Mean Flow Time. *Opsearch*. v. 30, n. 1, p. 34-41. 1993.

CALLAHAN, M.T. QUACKENBURH, D.G. e ROWINGS, J.E. *Construction Project Scheduling*. MacGraw-Hill. 1992.

CAO, J. e BEDWORTH, D.D. Flow Shop Scheduling In Serial Multi-Product Processes With Transfer And Set-Up Times. *International Journal of Production Research*, v. 30, n. 8, p. 1819-1830. 1992.

CARLIER, J. e PINSON, E. An Algorithm For Solving The Job-Shop Problem. *Management Science*. v. 35, n. 2, p. 164-176. 1989.

CARRENO, J.J. Economic Lot Scheduling For Multiple Products On Parallel Identical Processors. *Management Science*. v. 36, n. 3, p. 348-358. 1990.

CHAMBERS, R.J., CARRAWAY, R.L., LOWE, T.J e MORIN, T.L. Dominance And Decomposition Heuristics For Single Machine Scheduling. *Operations Research*. v. 39, n. 4, p. 639 - 648. 1991.

CHAN, K.C. e TANSRI, H. A Study Of Genetic Crossover Operations On The Facilities Layout Problem. *Computers Industrial Engineering*. v. 26, n. 3, p. 537-550. 1994.

CHANDRU, V., LEE, C.-Y. e UZSOY, R. Minimizing Total Completion Time On Batch Processing Machines. *International Journal of Production Research*. v. 31, n. 9, p. 2097-2121. 1993.

CHANG, T-CH. Network Resource Allocation Using An Expert System With Fuzzy Logic Reasoning. Ph.D. Dissertation. University of Berkeley. 1987.

CHANG, T-CH. Network Resource Allocation With Support Of A Fuzzy Expert System. *Journal Of Construction Engineering And Management*. v. 116, n. 2, p. 239-249. 1990.

CIB-90. International Council For Building Research Studies And Documentation. 1990. *Proceeding XIIth. International Congress CIB 90. V.2: Building Economic And Construction*. UTS/CIB, Sydney, Australia. 1990.

CHENG, C. e BULFIN R.L. Complexity Of Single Machine, Multi-Criteria Scheduling Problems. *European Journal of Operational Research*. v. 70, p. 115-125. 1993.

CHRYSSOLOURIS, G., DICKE, K., and LEE, M. On The Resources Allocation Problem. *International Journal of Production Research*. v. 31, n. 12, p. 2773-2795. 1992.

CHU, C. A Branch-And-Bound Algorithm To Minimize Total Flow Time With Unequal Release Dates. *Naval Research Logistics*. v. 39, p. 859-875. 1992a.

CHU, C. A Branch-And-Bound Algorithm To Minimize Total Tardiness With Different Release Dates. *Naval Research Logistics*. v. 39, p. 265-283. 1992.

CHU, C., FORTMANN, M.C. e PROTH, J.M. A Splitting-Up Approach To Simplify Job-Shop Scheduling Problems. *International Journal of Production Research*. v. 30, n. 4, p. 859-870. 1992.

COFFMAN, JR. E.G. Introduction To Deterministic Scheduling Theory. In: *Computer And Job-Shop Scheduling Theory*. Coffman, Jr.E.G. John Wiley & Sons. 1976.

COOPER, D.F. Heuristics For Scheduling Resource Constrained Projects: An Experimental Investigation. *Management Science*. v. 22, n. 11, p. 186-1184. 1976.

DAUZERE-PERES, S. The One-Machine Sequencing Problem With Dependent Jobs. *Computer and Industrial Engineering.*, v. 25, n. 1/44, p. 235-238. 1993.

DAUZERE-PERES, S. e LASSERRE, J-B. An Iterative Procedure For Lot Streaming In Job-Shop Scheduling. *Computers and Industrial Engineering*. v. 25, n. 1/4, p. 231-234. 1993a.

DAUZERE-PERES, S. e LASSERRE, J-B. A Modified Shifting Bottleneck Procedure For Job-Shop Scheduling. *International Journal of Production Research*. v. 31, n. 4, p. 923-932. 1993.

DAVIS, W.E. Project Scheduling Under Resource Constraints - Historical Review And Categorization Of Procedures. *AIIE Transactions*. v. 5, n. 4, p. 147-163. 1973.

DAVIS, W. E. Project Network Summary Measures Constrained-Resource Scheduling. *AIIE Transactions*. v. 7, n. 2, p. 132-142. 1973b.

DAVIS, W.E. Network: Resource Allocation. *Journal of Industrial Engineering*. v. 6, n. 4, p. 22-32. 1974.

DAVIS, E.W. e PATTERSON, J.H. A Comparison Of Heuristic And Optimum Solutions In Resource-Constrained Project Scheduling. *Management Science*. v. 21, n. 8, p. 944-955. 1975.

DAVIS, K.R., STAM, A. e GRZYBOWSKI, R.A. Resource Constrained Project Scheduling With Multiple Objectives: A Decision Support Approach. *Computers Operations Research*. v. 19, n. 7, p. 657-669. 1975.

DAVIS, J.S. e KANET, J.J. Single-Machine Scheduling With Early And Tardy Completion Costs. *Naval Research Logistics*. v. 40, p. 85-101. 1993.

DAVIS, L. Adapting Operator Probabilities in Genetic Algorithms. *Proceeding Of The Third International Conference On Genetic Algorithms*. San Mateo, CA. p. 61-69. 1989.

DAY, J.E e HOTTENSTEIN, M.P. Review Of Sequencing Research. *Naval Research Logistics Quarterly*. v. 17, p. 11-40. 1970,

DE, P., GHOSH, J.B. e WELLS, C.E. On The Minimization Of Time Variance With Bicriteria Extension. *Operations Research*. v. 40, n. 6, p. 1148-1155. 1992.

DE, P., GHOSH, J.B. e WELLS, C.E. Job Selection And Sequencing On A Single Machine In A Random Environment. *European Journal of Operational Research*. v. 70, p. 425-431. 1993.

DECKRO, R.F and HEBERT, J.E. Resource Constrained Project Crashing. *OMEGA Int. J. Of Management Science*. v. 17, n. 1, p. 69-79. 1989.

DE JONG, K.A. Analysis Of The Behavior Of A Class Of Genetic Adaptive Systems, Ph.D. Dissertation, Department of Computer and Communication Science, University of Michigan, Ann Arbor, MI. 1975.

DELL'AMICO, M. e TRUBIAN, M. Applying Tabu Search To The Job-Shop Scheduling Problem. *Annals of Operations Research*. v. 41, p. 231-252. 1993.

DEMEULEMEESTER, E. e HERROELEN, W. A Branch-And-Bound Procedure For The Multiple Resource-Constrained Project Scheduling Problem. *Management Science*. v. 38, n. 12, p. 1803-1818. 1992.

DOBSON, G. e KARMARKAR, U.S. Simultaneous Resource Scheduling To Minimize Weighted Flow Times. *Operations Research*. v. 37, n. 4, p. 592-600. 1989.

DOBSON, G., KARMARKAR, U.S. e RUMMEL, J.L. Batching To Minimize Flow Times On Parallel Heterogeneous Machines. *Management Science*. v. 35, n. 5, p. 607-613. 1989.

DOCTOR, S.R., CAVALIER, T.M. e EGBE:U, P.J. Scheduling For Machining And Assembly In A Job-Shop Environment. *International Journal of Production Research*. v. 31, n. 6, p. 1275-1297. 1993.

DONDETI, V.T. e EMMONS, H. Fixed Job Scheduling With Two Types Of Processors. *Operations Research*. v. 40, supp.1, p. S76. 1992.

DUMOND, J. e MABERT, V.A. Evaluating Project Scheduling And Due Date Assignment Procedures: An Experimental Analysis. *Management Science*. v. 34, n. 1, p. 101-118. 1988.

DREXEL, A. e GRUENEWALD, J. Nonpreemptive Multi-Mode Resource-Constrained Project Scheduling. *IIE Transaction*, v. 25, n. 5, p. 74-81. 1993.

ELMAGHRABY, S.E. The Machine Sequencing Problem - Review And Extensions. *Naval Research Logistics Quarterly.* v. 15, p. 205-232. 1968.

ELMAGHRABY, S.E. e PULAT, P.S. Optimal Project Compression With Due-Dated Events. *Naval Research Logistic Quaterly.* v. 26, n. 2, p. 331-348. 1979.

ELMAGHRABY, S.E. e HERROELEN, W. S. On The Measurement Of Complexity In Activity Networks. *European Journal Of Operational Research.* v. 5, p. 223-234. 1980.

ELSAYED, E.A. e NASR, N.Z. Heuristic For Resource-Constrained Scheduling. *International Journal of Production Research.* v. 24, n. 2, p. 299-310. 1986.

EPSTEIN, S., WILAMOWSKY, Y. e DICKMAN, B. Deterministic Multiprocessor Scheduling With Multiple Objectives. *Computers Operations Research.* v. 19, n. 8, p. 743-749. 1992.

FALKENAUER, E. e BOUFFOUIX, S. A Genetic Algorithm For Job Shop. *Proceeding Of The 1991 IEEE: International Conference On Robotics And Automation.* p. 824-829. 1991.

FALKENAUER, E. e DELCHAMBRE, A. A Genetic Algorithm For Bin Packing And Line Balancing. *Proceeding Of The 1992 IEEE: International Conference On Robotics And Automation.* p. 1186-1192. 1992.

FANG, H-L, ROSS, P. e CORNE, D. A Promising Genetic Algorithm Approach To Job-Shop Scheduling, Rescheduling, And Open-Shop Scheduling Problems. *Proceeding Of The Fifth International Conference On Genetic Algorithms.* Urbana, IL. p. 375-382. 1993.

FAISON, T. *Borland C++ 4 Object-Oriented Programming.* Third Edition, SAMS Publishing. 1994.

FLAMIG, B. *Turbo C++: Um Guia Para Auto-Aprendizado.* Livros Técnicos e Científicos Editora. Rio de Janeiro. 1992.

FORMOSO, C. e BRANDON, P.S. Expert Systems In The Construction Industry. *Annals Of IX ENEGEP*, Brazil. v. 2, p. 174-186. 1989.

FORST, F.G. Minimizing Total Expected Costs In The Two-Machine, Stochastic Flow Shop. *Operations Research Letters.* v. 2, p. 58-61. 1983.

FORST, F.G. A Review Of The Static, Stochastic Job Sequencing Literature. *Opsearch.* v. 21, p. 127-144. 1984.

FRANÇA, P.M., GENDREAU, M., LAPORTE, G. e MÜLLER, F.M. A Composite Heuristic For The Identical Parallel Machine Scheduling Problem With Minimum Makespan Objective. *Computer Operations Research.* v. 21, n. 2, p. 205-210. 1994.

FRENCH, S. *Sequencing And Scheduling.* John Wiley, New York. 1986

GALLEGO, G. Scheduling The Production Of Several Items With Random Demands In A Single Facility. *Management Science.* v. 36, n. 12, p. 1579-1592. 1990.

GAREY, M.R., JOHNSON, D.S. e SETHI, R. The Complexity Of Flowshop And Jobshop Scheduling, *Mathematics of Operations Research.* v. 1, p. 117-129. 1976.

GARVEY, A.J. e LESSER, V.R. Design-Time Real-Time Scheduling. *IEEE Transaction On Systems, Man, And Cybernetics.* v. 23, n. 6, p. 1491-1502. 1993.

GAUTHIER, F.A.O. Programação Da Produção: Uma Abordagem Utilizando Algoritmos Genéticos. Tese De Doutorado, Universidade Federal De Santa Catarina. 1993.

GLAZEBROOK, K.D. Single-Machine Scheduling Of Stochastic Jobs Subject To Deterioration Or Delay. *Naval Research Logistics.* v. 39, p. 613-633. 1992.

GODIN, V.B. Interactive Scheduling: Historical Survey And State Of The Art. *AIIE Trans.* v. 10, p. 331-337. 1978.

GOLDBERG, D.E. Sizing Population For Serial And Parallel Genetic Algorithms. *Proceeding Of The Third International Conference On Genetic Algorithms.* San Mateo, CA. p. 70-79. 1989.

GOLDBERG, D.E.. *Genetic Algorithms In Search, Optimization, And Machine Learning.* Addison-Wesley, Reading, Mass. 1989

GOYAL, S.K. e SRISKANDARAJAH, C. No-Wait Shop Scheduling: Computational Complexity And Apprcximate Algorithms. *Opsearch.* v. 25, p. 220-244. 1988.

GORDON, V.S. A Note On Optimal Assignment Of Slack Due-Dates In Single-Machine Scheduling. *European Journal of Operational Research.* v. 70, p. 311-315. 1993.

GRAHAM, R.L., LAWLER, E.L., LENSTRA, J.K. e RINNOOY KAN, A.H.G. Optimization And Approximation In Deterministic Sequencing And Scheduling: A Survey. *Annals of Discrete Mathematics.* v. 5, p. 287-326. 1979.

GRAVES, S.G. A Review Of Production Scheduling. *Operations Research.* v. 29, p. 646-675. 1981.

GREFENSTETTE, J. J. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transaction On Systems, Man, And Cybernetic.* v. 16, n. 1, p. 122-128. 1989.

GUINET, A. Scheduling Sequence-Dependent Jobs On Identical Parallel Machines To Minimize Completion Time Criteria. *International Journal of Production Research.* v. 31, n. 7, p. 1579-1594. 1993.

GUPTA, J.N.D. e TUNC, E.A. Scheduling For A Two-Stage Hybrid Flowshop With Parallel Machines At The Second Stage. *International Journal of Production Research.* v. 29, n. 7, 1489-1502. 1991.

GUPTA, J.N.D. e POTTS, C.N. Editorial. *European Journal of Operational Research.* v. 70, p. 269-271. 1993.

GUPTA, M.C., GUPTA, Y.P. e EVANS, G.W., Operations Planning And Scheduling Problems In Advanced Manufacturing Systems. *International Journal of Production Research.* v. 31, n.4, p. 869-900. 1993.

GUPTA, M.C., GUPTA, Y.P. e BECTOR, C.R. Minimizing The Flow-Time Variance In Single-Machine Systems. *Journal of Operational Research Society.* v. 41, n. 8, p. 767-779. 1990.

GUPTA, M.C., GUPTA, Y.P. e KUMAR, A. Minimizing Flow Time Variance In A Single Machine System Using Genetic Algorithms. *European Journal of Operational Research.* v. 70, p. 289-303. 1993.

HALL, N.G. e POSNER, M.E. Earliness-Tardiness Scheduling Problems, I: Weighted Deviation Of Completion Times About A Common Due Date. *Operations Research.* v. 39, n. 5, 836-846. 1991.

HALL, N.G., KUBIAK, W e SETHI, S.P. Earliness-Tardiness Scheduling Problems, II: Deviation Of Completion Times About A Restrictive Common Due Date. *Operations Research.* v. 39, n. 5, p. 836-847. 1991.

HARRIS, R.B. Packing Method For Resource Leveling (PACK). *Journal Of Construction Engineering And Management.* v. 116, n. 2, p. 331-350. 1990.

HASSAN M.H. e AYYUB, B.M. A Fuzzy Controller For Construction Activities. *Fuzzy Sets And Systems.* v. 56, p. 253-271. 1993.

HENDRY, L.C. e KINGSMAN, B.G. Production Planning Systems And Their Applicability To Make-To-Order Companies. *European Journal of Operational Research.* v. 40, p.1-15. 1989.

HERRBACH, L.A. e LEUNG, J.Y.-T. Preemptive Scheduling Of Equal Length Jobs On Two Machine To Minimize Mean Flow Time. *Operations Research.* v. 38, n. 3), 487-494. 1990.

HERRMANN, J.W. e LEE, C. On Scheduling To Minimize Earliness-Tardiness And Batch Delivery Costs With A Common Due Date. *European Journal of Operational Research.* v. 70, p. 272-288. 1992.

HERROELEN, W.S. Resource Constrained Project Scheduling - The State Of The Art. *Operation Research Quarterly.* v. 23, n. 3, p. 261-275. 1972.

HO, J.C., e CHANG, Y.-L. Heuristic For Minimizing Mean Tardiness For *M* Parallel Machines. *Naval Research Logistics.* v. 38, p. 367-381. 1991.

HOLSENBACK, J.E. e RUSELL, R.M. A Heuristic Algorithm For The Sequencing On One Machine To Minimize Total Tardiness. *Journal of Operational Research Society.* v. 43, n. 1, p. 53-62. 1991.

HOLLAND, J.H. *Adaptation In Natural And Artificial Systems.* University Of Michigan Press, Ann Arbor. 1975.

HOLLOWAY, C.A., NELSON, R.T. e SURAPHONGSCHAI, V. Comparison Of A Multi-Pass Heuristic Decomposition Procedure With Other Resource Constrained Project Scheduling Procedures. *Management Science.* v. 25, n. 9, p. 862-872. 1979.

IBARAKI, T. Combinatorial Optimization Problems And Their Complexity, In: Enumerative Approaches To Combinatorial Optimization - Part I. *Annals of Operations Research*, v. 10. 1987.

IBRAHIM, A. e THOMAS, L.C. A Two-Product, Single-Machine, Storage-Constrained Production Problem. *Journal of Operational Research Society.* v. 42, n. 9, p. 785-791. 1991.

JEFFCOAT, D.E. e BULFIN, R.L. Simulated Annealing For Resource-Constrained Scheduling. *European Journal of Operational Research.* v. 70. p. 43-51. 1993.

JOG, P., SUH, J.Y. e GUCHT, D.V. The Effect Of Population Size, Heuristic Crossover And Local Improvement On A Genetic Algorithm For A Traveling Salesman Problem. *Proceeding Of The Third International Conference On Genetic Algorithms.* San Mateo, CA. p. 110-115. 1989.

JOHNSON, S.M. Optimal Two And Three-Stage Production Schedules With Set-Up Times Included. *Naval Research Logistics Quarterly*. v. 1, p. 61-68. 1954.

JOHNSON, G.A. e SCHOU, C.D. Expediting Projects In PERT With Stochastic Time Estimates. *Project Management Journal*. v. XXI, n. 2, p. 29-33. 1990.

JONES, R.A. Resource Scheduling: A Monte Carlo Approach to CPM. *Proceeding of CIB W-65*. p. 422-427. 1984.

KAMOUN, H. e SRISKANDARAJAH, C. The Complexity Of Scheduling Jobs In Repetitive Manufacturing Systems. European *Journal of Operational Research*. v. 70, p. 350-364. 1993.

KARABATI, S., KOUVELIS, P. e KIRAN, A.S. Games, Critical Paths And Assignment Problems In Permutation Flow Shops And Cyclic Scheduling Flow Line Environments. *Journal of the Operational Research Society*. v. 43, n. 3, p. 241-258. 1992.

KARABATI, S., KOUVELIS, P. Completion Times Performance Criterion. *Naval Research Logistics*. v. 40, p. 843-862. 1993.

KHATTAB, M.M. e CHOOBINEH, F. A New Approach For Project Scheduling With A Limited Resource. *International Journal of Production Research*. v. 29, n. 1, p. 185-198. 1991.

KIM, Y.D. A New Branch And Bound Algorithm For Minimizing Mean Tardiness In Two-Machine Flowshops. *Computers Operations Research*. v. 20, n. 4, p. 391-401. 1993.

KIM, G. e LEE, C.S.G. An Evolutionary Approach To The Job-Shop Scheduling Problem. *Proceeding Of The 1994 IEEE International Conference On Robotics And Automation*. San Diego, CA. p. 501-506. 1994.

KIM, S.Y. e LEACHMAN, R.C. Multi-Project Scheduling With Explicit Lateness Costs. *IIE Transactions*. v. 25, n. 2, p. 34-44. 1993.

KING, J.R. e SPACHIS, A.S. Scheduling: Bibliography & Review. *International Journal of Physical Distribution and Materials Management*. v. 10, n. 3, p. 105-132. 1980.

KIRAN, A.S. e UNAL, A.T. A Single-Machine Problem With Multiple Criteria. *Naval Research Logistics*. v. 38, p. 721-727. 1991.

KOLEN, A.W.J. e KROON, L.G. On The Computational Complexity Of (Maximum) Class Scheduling. *European Journal of Operational Research*. v. 54, p. 23-38. 1991

KOSTEM, C. e MAHER, M.L. Expert Systems In Civil Engineering. *Proceeding Of A Symposium*, Seattle, WA, USA. 1986.

KULKARNI, V.G. e CHIMENTO Jr., P.F. Optimal Scheduling Of Exponential Task With In-Tree Precedence Constraints On Two Parallel Processors Subject To Failure And Repair. *Operations Research*. v. 40, supp. n. 2, p. S263. 1992.

KURTULUS, I. e DAVIS, E.W. Multi-Project Scheduling: Categorization Of Heuristic Rules Performance. *Management Science*. v. 28, n. 2, p. 161-172. 1982.

KURTULUS , I. S e NARULA, S.C. Multi-Project: Analysis Of Project Performance. *IIE Transactions*. v. 17, n. 1, p. 58-66. 1985.

LAWRENCE, S.R. Scheduling A Single Machine To Maximize Net Present Value. *International Journal of Production Research*. v. 29, n. 6, p. 1141-1160. 1991.

LAWTON, J. Genetic Algorithms For Scheduling Optimization. *AI Expert*. v. 7, n. 5, p. 22-26. 1992.

LEACHMAN, R.C. e GASCON, A. A Heuristic Scheduling Policy For Multi-Item, Single-Machine Production Systems With Time-Varying, Stochastic Demands. *Management Science*. v. 34, n. 3, p. 377- 390. 1988.

LEACHMAN, R.C., e KIM, S. A Revised Critical Path Method For Networks Including Both Overlap Relationships And Variable-Duration Activities. *European Journal Of Operational Research*. v. 64, p. 229-248. 1993.

LEACHMAN, R.C., DINCERLER, A. e KIM, S. Resource Constrained Scheduling Of Project With Variable Intensity Activities. *IIE Transactions*. v. 22, n. 1, 31-40. 1990.

LEE, I. A Worst-Case Performance Of The Shortest-Processing-Time Heuristic For Single Machine Scheduling. *Journal of The Operational Research Society*. v. 42, n. 10, p. 895-901. 1991.

LEE,Y H. e KIM, S. Neural Network Application For Scheduling Jobs On Parallel Machines. *Computer and Industrial Engineering*. v. 25, n.1/4, p. 227-230. 1993.

LEE, C.Y., CHENG, T.C.E. e LIN, B.M.T. Minimizing The Makespan In The 3-Machine Assembly-Type Flowshop Scheduling Problem. *Management Science*. v. 39, n. 5, p. 616-625. 1993.

LEON, V.J. e WU S.D. On Scheduling With Ready-Times, Due-Dates And Vacations. *Naval Research Logistics*. v. 39, p. 53-65. 1992.

LESNTRA, J.K. *Sequencing By Enumerative Methods*. Mathematical Center Tracts 69, Amsterdam. 1977.

LENSTRA, J.K. e RINNOOY KAN, A.H.G. Complexity Of Scheduling Under Precedence Constraints, *Operations Research*. v. 26, n. 1, p. 22-35. 1978

LENSTRA, J.K., RINNOOY KAN, A.H.G., e BRUCKER, P. Complexity Of Machine Scheduling Problems, *Annals of Discrete Mathematics*. v. 1, p. 343-362. 1977

LEVITT, R. E. e KUNZ, J.C. Using Knowledge Of Construction And Project Management For Automated Schedule Updating. *Project Management Journal*. v. 16, n. 5, p. 57-76. 1985.

LI, K.Y., e WILLIS, R.J. An Iterative Scheduling Technique For Resource-Constrained Project Scheduling. *European Journal Of Operational Research*. v. 56, p. 370-379. 1992.

LIAO, C.J. e YOU, C.T. An Improved Formulation For The Job-Shop Scheduling Problem. *Journal of the Operational Research*. v. 43, n. 11, p. 1047-1054. 1992.

LIU, J. e MacCARTHY, B.L. Effective Heuristics For The Single Machine Sequencing Problem With Ready Times. *International Journal of Production Research*. v. 29, n. 8, p. 1521-1533. 1990.

LOCKYER, K.G. *An Introduction To Critical Path Analysis*. 3rd edition, Pitman, London. 1981.

MACCARTHY B.L e LIU J. Addressing The Gap In Scheduling Research: A Review Of Optimization And Heuristic Methods In Production Scheduling. *International Journal of Production Research.* v. 31, n. 1, p. 59-79. 1993.

MASON, A.J. e ANDERSON, E.J. Minimizing Flow Time On A Single Machine With Job Classes Setup Times. *Naval Research Logistics.* v. 38, p. 333-350. 1991.

MATSSURA, H., TSUBONE, H. e KANEZASHI, M. Sequencing, Dispatching, And Switching In A Dynamic Manufacturing Environment. *International Journal of Production Research.* v. 31, n. 7, p. 1671-1688. 1993

MÜLLER-MERBACH, H. Heuristic And Their Design: A Survey. *European Journal of Operational Research.* v. 8, p. 1-23. 1981.

MINKARAH, I. e AHMAD, I. Expert Systems As Construction Management Tools. *Journal Of Management In Engineering.* v. 5, n. 2, p. 155-163. 1989.

MITTENTHAL, J., RAGHAVACHARI, M. e RANAS, A. A Hybrid Simulated Annealing Approach For Single Machine Scheduling Problems With Non-Regular Penalty Functions. *Computer Operational Research.* v. 20, n. 2, p. 103-111. 1993.

MODER, J.J., PHILLIPS, C.R. e DAVIS, *E.W. Project Management With CPM, PERT And Precedence Diagramming.* 3rd ed., Reinhold, New York. 1983.

MOHAN, S. Expert Systems Application In Construction Management And Engineering. *Journal Of Construction Engineering And Management.* v. 116, n. 1, p. 87-99. 1990.

MOHANTHY, R.P. e SIDDIQ, M.K. Multiple Projects Multiple Resources-Constrained Scheduling: Some Studies. *International Journal of Production Research.* v. 27, n. 2, p. 261-280. 1989.

MONMA C.L. e POTTS C.N. On The Complexity Of Scheduling Batch Setup Times. *Operations Research.* v. 37, p. 798-804. 1989.

MOSELHI, O. e NICHOLAS, M.J. Hybrid Expert System For Construction Planning And Scheduling. *Journal Of Construction Engineering And Management.* v. 116, n. 2, p. 221-238. 1990.

MOSELHI, O. e LORTERAPONG, P. Near Optimal Solution For Resource-Constrained Scheduling Problems. *Construction Management and Economics.* v. 11, p. 293-303. 1993.

MOSHELI, O. Scheduling Of Multiple-Story Buildings In A Constraint Environment. *Proceeding Of The Fifth International Conference.* (V-ICCCBE). Anaheim, CA. p. 1822-1829. 1993a.

MOSHELI, O. An OOP Model For Scheduling Of Repetitive Projects. *Proceeding Of The Fifth International Conference.* (V-ICCCBE). Anaheim, CA. p. 939-946. 1993b.

MUSTAFA, M.A. e MURPHREE, E.L. A Multicriteria Decision Support Approach For Project Compression. *Project Management Journal.* v. XX, n. 2, p. 29-34. 1989.

NAKANO, R. e YAMADA, T. Conventional Genetic Algorithm For Job Shop Problems. *Proc. 4th. Int. Conf. on Genetic Algorithms*, San Diego, CA. p. 474-479. 1991.

NANDKEOLYAR, U., AHMED, M.U. e SUNDARARAGHAVEN, P.S. Dynamic Single-Machine-Weighted Absolute Deviation Problem: Predictive Heuristics And Evaluation. *International Journal of Production Research.* v. 31, n. 6, p. 1453-1466. 1993.

NORBIS, M.I . e SMITH, J.M. Two Level Heuristic For The Resource Constrained Scheduling Problem. *International Journal of Production Research.* v. 24, n. 5. p. 1203-1219. 1986.

NORONHA, S.J. e SARMA, V.V.S. Knowledge-Based Approaches For Scheduling Problems: A Survey. *IEEE Transaction On Knowledge And Data Engineering.* v. 3, n. 2, p. 160-171. 1991.

NOWICKI, E. An Approximation Algorithm For The *M*-Machine Permutation Flow Shop Scheduling With Controllable Processing Times. *European Journal of Operational Research.* v. 70, p. 342-349. 1993.

OGUZ , O. e BALA, H. A Comparative Study Of Computational Procedures For The Resource Constrained Project Scheduling Problem. *European Journal of Operational Research.* v. 72, p. 406-416. 1994.

OLIVER, I.M., SMITH, D.J., e HOLLAND, J.R.C. A Study Of Permutation Crossover Operation On The Traveling Salesman Problem. *Proceeding Of The 2nd. International Conference Of Genetic Algorithms.* p. 224-230. 1987.

PANWALKER, S.S. e ISKANDER, W. A Survey Of Scheduling Rules. *Operations Research.* v. 25, p. 45-61. 1977.

PANWALKER, S.S., SMITH, M.L. e KOULAMAS, C.P. A Heuristic For Single Machine Tardiness Problem. *European Journal of Operational Research.* v. 70, p. 304-310. 1993.

PATTERSON, JH. Alternate Methods Of Project Scheduling With Limited Resources. *Naval Research Logistic Quarterly.* v. 20, n. 4, p. 767-784. 1973.

PATTERSON, J.H. Project Scheduling: The Effect Of Problem Structure On Heuristic Performance. *Naval Research Logistic Quarterly.* v. 23, n. 1, p. 95-123. 1976.

PATTERSON, J.H. A Comparison Of Exact Approaches For Solving The Multiple Constrained Resource Project Scheduling Problem. *Management Science.* v. 30, n. 7, p. 854-867. 1984.

PATTERSON, J.H., TALBOT, F.B., SLOWINSKI, R. e WEGLARZ, J. Computational Experience With A Backtracking Algorithm For Solving A General Class Of Precedence And Resource-Constrained Scheduling Problems. *European Journal of Operational Research.* v. 49. p. 68-79. 1990.

PHAM, D.T. e PHAM, P.T.N. Expert Systems: A Review. In: *Expert Systems In Engineering.* IFS Pub/Springer-Verlag. London. p. 3-18. 1988.

POTTS, C.N. e VAN WASSENHOVE, L.N.. Algorithms For Scheduling A Single Machine For Minimizing The Weighted Number Of Late Jobs. *Management Science.* v. 34, n. 7, p. 843-858. 1988

POTTS, C.N. e VAN WASSENHOVE, L.N. Single Machine Scheduling To Minimize Total Late Work. *Operations Research.* v. 40, n. 3, p. 586-595. 1992.

PRADO, D. *Administração De Projetos Com PERT/CPM.* Rio De Janeiro. Livros Técnicos E Científicos Editora S.A. 1984.

PRENTIS, EL. Master Project Planning: Scope, Time And Cost. *Project Management Journal*, v. XX, n. 1, p. 24-30. 1989.

PRITSKER, A.A., WATTERS, L.J. e WOLFE, D.M. Multi-Project Scheduling With Limited Resources: A Zero-One Programming Approach. *Management Science*. v. 24, n. 22, p. 1163-1174. 1969.

PROUST, C., GUPTA, J.N.D. e DESCHAMPS, V. Flowshop Scheduling With Set-Up, Processing And Removal Times Separated. *International Journal of Production Research*. v. 29, n. 3, p. 479-493. 1991.

QUINLAN, J.R. Application Of Expert Systems. *Proceeding Of The Second Australian Conference.* Turing Inst. Press/Addison-Wesley. 1987.

RAJENDRAN, C. Two-Stage Flowshop Scheduling Problem With Bicriteria. *Journal of the Operational Research Society*. v. 43, n. 9, p. 871-884. 1992.

RAJGOPAL, J. e BIDANDA, B. On Scheduling Parallel Machines With Two Setup Classes. *International Journal of Production Research*. v. 29, n. 12, p. 2243-2458. 1991.

RASFORF, W.J. e ABUDAYYED, O.Y. Cost And Schedule Control Integration: Issues And Needs. *Journal Of Construction Engineering And Management*. v. 117, n. 3, p. 486-502. 1991.

REEVES, C.R. Genetic Algorithms. In: *Modern Heuristic Techniques For Combinatorial Problems*. John Wiley & Sons, Inc. New York, N.Y. 1993.

RINNOY KAN, H.H.G. *Machine Scheduling Problems: Classification, Complexity And Computations.* The Hague: Nijhoff. 1976.

RODAMMER, F.A. e WHITE, Jr. K.P. A Recent Survey Of Production Scheduling. *IEEE Transaction on Systems, Man and Cybernetics*. v. 18, n. 6, p. 841-851. 1988.

SAMPSON, S.E. e WEISS, E.N. Local Search Techniques For The Generalized Resource Constrained Project Scheduling Problem. *Naval Research Logistics*. v. 40, p. 665-675. 1993.

SEIBERT, J.E e EVANS, G.W. Time-Constrained Resource Leveling. *Journal Of Construction Engineering And Management*. v. 117, n. 3, p. 503-520. 1991.

SEN, T. e BORAH, B.N. On The Single-Machine Scheduling Problem With Tardiness Penalties. *Journal of Operations Research Society*. v. 42, n. 8, p. 695-702. 1991.

SCHAFFER, J.D., CARUANA, R.A., ESHELMAN, L.J. e DAS, R. A Study Of Control Parameters Affecting On-line Performance Of Genetic Algorithms For Function Optimization., *Proceeding Of The Third International Conference On Genetic Algorithms*. San Mateo, CA. p. 51-60. 1989.

SO, K.C. Some Heuristics For Scheduling Jobs On Parallel Machines With Setups. *Management Science*. v. 36, n. 4, p. 467 - 475. 1990.

SLOWINSKI, R. Two Approaches To The Problem Of Resource Allocation Among Project Activities - A Comparative Study . *Journal Of The Operational Research Society*. v. 31, n. 8, p. 711-723. 1980.

SLOWINSKI, R. Multiobjective Network Scheduling With Efficient Use Of Renewable And Nonrenewable Resources. *European Journal Of Operational Research*. v. 7, p. 265-273. 1981.

SMITH, R., PERELSON, A.S. e FORREST, S. Searching For Diverse, Cooperative Population With Genetic Algorithms. *Evolutionary Computation*. v. 1, n. 2, p. 127-149. 1993.

SPONSLER, J.L. Genetic Algorithms Applied To The Scheduling Of The Hubble Space Telescope. *Telematics And Informatics*. v. 6, n. 3/4, p. 181-190. 1989.

SRINIVAS, M. e PATNAIK, L. M. Adaptative Probabilities Of Crossover And Mutation In Genetic Algorithms. *IEEE Transaction On Systems, Man And Cybernetics*. v. 24, n. 4, p. 656-667. 1994.

STORER, H.R., WU, S,W. e VACCARI, R. New Search Spaces For Sequencing Problems With Application To Job Shop Scheduling. *Management Science*. v. 38, n. 10, p. 1495-1509. 1992.

SU, Z.S. e SEVCIK, K.C. A Combinatorial Approach To Dynamic Scheduling Problems. *Operations Research*. v. 26, n. 5, p. 836-844. 1978.

SÜER, G.A. e BÁEZ. Minimizing The Number Of Tardy Jobs In Identical Machine Scheduling. *Computer and Industrial Engineering* v. 25, n. 1/4, p. 243-246. 1993.

SUNG. C.S. e JOO, U.G. A Single Machine Scheduling Problem With Earliness/Tardiness And Starting Time Penalties Under A Common Due Date. *Computers and Operational Research*. v. 19, n. 8, p. 757-766. 1992.

SYSWERDA, G., e PALMUCCI, J. The Application Of Genetic Algorithms To Resource Scheduling. *Proceeding Of The Fourth International Conference On Genetic Algorithms*. San Diego, CA. p. 502-508. 1991.

SZWARC, W. Adjacent Orderings In Single-Machine Scheduling With Earliness And Tardiness Penalties. *Naval Research Logistics*. v. 40, p. 229-243. 1993.

SZWARC, W., POSNER, M.E. e LIU, J.L. The Single Machine Problem With A Quadratic Cost Function Of Completion Time. *Management Science*. v. 34, n. 12, p. 1480-1488. 1988.

SZWARC, W. e LIU, J.J. Weighted Tardiness Single Machine Scheduling With Proportional Weights. *Management Science*. v. 39, n. 5, p. 626-632. 1993.

TALBOT, F.B. Resource-Constrained Project Scheduling With Time-Resource Tradeoffs: The Nonpreemptive Case. *Management Science*. v. 28, n. 10, p. 1197-1210. 1982.

TAVARES, L.V. A Multi-Stage Non-Deterministic Model For Project Scheduling Under Resources Constraints. *European Journal of Operational Research*. v. 49, p. 92-101. 1990.

THOMASEN, O.B. e BUTTERFIELD, L. Combining Risk Management And Resource Optimization In Project Management Software. *Cost Engineering*. v. 35, n. 8, p. 19-25. 1993.

TSUBAKITANI, S. e DECKRO, R.F. A Heuristic For Multi-Project Scheduling With Limited Resources In The Housing Industry. *European Journal of Operational Research*. v. 49. p. 80-91. 1990.

TSUJIMURA, Y., PARK, S.H., CHANG, I.S. e GEN, M. An Effective Method For Solving Flow Shop Scheduling Problems With Fuzzy Processing Times. *Computers and Industrial Engineering*. v. 25, n. 1/4, p. 239-242. 1993.

ULLMAN, J.D., NP-Complete Scheduling Problems. *Journal of Computing System Science*. v. 10, p. 384-393. 1975

ULLMAN, J.D. Complexity Of Sequencing Problems. In: *Computer And Job-Shop Scheduling Theory*. John Wiley & Sons. 1976.

VAN DE VEL, H. e SHIJIE, S. An Application Of The Bin-Packing Technique To Job Scheduling On Uniform Processors. *Journal of The Operational Research Society*. v. 42, n. 2, p. 169-172. 1991.

VAN LAARHOVEN, P.J.M., AARTS, E.H.L. e LENSTRA, J.K. Job Shop Scheduling By Simulated Annealing. *Operations Research*. v. 40, n. 1, p. 113-125. 1992.

VEMPATI, V.S., CHEN, C.-L. e BULLINGTON, S.F. An Effective Heuristic For Flow Shop Problems With Total Flow Time As Criterion. *Computers and Industrial Engineering*. v. 5, n. 1/4, p. 219-222. 1993.

VICKSON, R.G. e ALFREDSSON, B.E. Two And Three Machine Flow Shop Scheduling Problems With Equal Sized Transfer Batches. *International Journal of Production Research*, v. 30, n. 7, p. 1551-1574. 1992.

WAGNEUR, E. e SRISKANDARAJAH, C. The Two-Machine Permutation Flow Shop With State-Dependent Processing Times. *Naval Research Logistics*. v. 40, p. 697-717. 1993.

WEBSTER, S. New Bounds For The Identical Parallel Processor Weighted Flow Time Problem. *Management Science*. v. 38, n. 1, p. 124-136. 1992

WEBSTER, S.T. A Priority Rule For Minimizing Weighted Flow Time In A Class Of Parallel Machine Scheduling Problems. *European Journal of Operational Research*. v. 70, p. 327-334. 1993.

WEGLARZ, J. Project Scheduling With Discrete And Continuous Resources. *IEEE Transaction On Systems, Man, And Cybernetics*. v. SMC-9, n. 10. p. 644-650. 1979.

WEGLARZ, J. Project Scheduling With Continuously-Divisible, Doubly Constrained Resources. *Management Science*. v. 27, n. 9. p. 1040-053. 1981.

WERNER, F. On The Heuristic Solution Of The Permutation Flow Shop Problem By Path Algorithms. *Computers Operations Research*. v. 20, n. 7, p. 707-722. 1993.

WIDMER, M. Job Shop Scheduling With Tooling Constraints: A Tabu Search Approach. *Journal of the Operational Research Society*. v. 42, n. 1, p. 75-82. 1991.

WIDMER, M. e HERTZ, A. A New Method For The Flow Sequencing Problem. *European Journal of Operational Research*. v. 41, p. 186-193. 1989.

WIEST, J.D. A Heuristic Model For Scheduling Large Projects With Limited Resources. *Management Science*. v. 3, n. 6, p. B359-B377. 1967.

WOODWORTH, B.M. A Statistical Evaluation Of The Impact Of Limited Resources On Project Scheduling. *Cost Engineering*. v. 35, n. 2, p. 25-32. 1993.

WU, S.D., STORES, R.H. and CHANG, P. One-Machine Rescheduling Heuristic With Efficiency And Stability As Criteria. *Computer Operational Research*. v. 20, n. 1, p. 1-14. 1993.

WU, R.W-K. E HADIPRIONO, F.C. Fuzzy Modus Ponens Deduction Technique For Construction Scheduling. *Journal Of Construction Engineering And Management*. v. 120, n. 1, p. 162-179. 1994.

ZHENG, W.-X., NAGASAWA, H. e NISHIYAMA, N. Single-Machine Scheduling For Minimizing Total Cost With Identical, Asymmetrical Earliness And Tardiness Penalties. *International Journal of Production Research*. v. 31, n. 7, p. 1611-1620. 1993.

ANEXO I

Dados Para Os Problemas Teste

NOTA

[1] Os primeiros 16 problemas são parte de um conjunto de 30 exemplos, gentilmente fornecidos pelo Professor Adedeji B. Badiru, diretor do Expert Systems Laboratory, School of Industrial Engineeering, University of Oklahoma, Norman OK, USA. Os 9 projetos restantes fazem parte do conjunto de problemas testes cedidos pelo Professor Raymond Li do Department of Bussiness Systems of the Monash University, Clayton, Australia

[2] Todos os projetos foram adaptados de forma a considerar o caso de múltiplos modos. O modo de execução indicado com a letra *a* corresponde à combinação recursos-duração proposta pelos autores originais dos exemplos.

PROBLEMA 1

FONTE ORIGINAL[1]: Whitehouse E Wechsler

RECURSO	**R1**	**R2**	**R3**
DISPONIBILIDADE	*4*	*5*	*2*

ATIVIDADE	**MODO**[2]	**DURAÇÃO**	**DEMANDA DE RECURSOS**			**ANTECESSORAS**
			R1	**R2**	**R3**	
1	a	5	3	-	1	-
	b	6	2	1	1	
	c	7	2	-	2	
2	a	3	1	4	1	1
	b	5	1	3	1	
3	a	7	2	3	1	2
	b	8	2	2	1	
	c	10	1	2	1	
4	a	15	2	1	-	1
	b	17	1	1	-	
5	a	3	2	-	1	1
	b	3	2	2	-	
6	a	7	1	3	2	1
	b	8	2	3	1	
	c	7	3	4		
7	a	3	4	2	1	1
	b	4	3	2	1	
	c	6	2	3	-	
8	a	10	1	2	1	3, 5
	b	12	1	1	1	
9	a	5	1	3	1	3, 4
	b	6	1	2	1	
10	a	11	2	-	1	3,4
	b	13	1	-	1	
	c	11	1	2	1	
11	a	9	3	1	1	8
	b	11	2	1	1	
12	a	24	1	1	1	6, 7, 9, 10, 11
13	a	12	2	1	1	7, 9, 10, 11
	b	14	1	1	1	
14	a	3	-	4	1	13
	b	3	1	2	1	
	c	5	-	3	1	
15	a	10	1	3	-	9, 10, 11
	b	12	1	2	-	
	c	10	1	1	1	
16	a	10	1	2	1	9, 10, 11
	b	12	1	1	1	
17	a	15	2	2	1	15, 16
	b	16	1	2	1	
	c	18	1	1	1	
18	a	5	3	3	1	17
	b	6	2	3	1	
	c	8	2	2	1	
19	a	10	2	1	1	8
	b	12	1	1	1	
20	a	8	2	1	1	3
	b	11	1	1	1	
21	a	12	2	1	1	3
	b	12	1	3	1	
	c	14	1	1	1	
22	a	8	2	4	-	20, 21
	b	10	2	3	-	
	c	8	2	2	1	
23	a	2	1	-	-	12, 14, 18, 19, 22
24		5	4	1	1	23

PROBLEMA 2

FONTE ORIGINAL[1]: Moodie e Young

RECURSO	R1	R2	R3	R4
DISPONIBILIDADE	*2*	*2*	*1*	*3*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS				ANTECESSORAS
			R1	R2	R3	R4	
1	a	4	1	2	1	1	-
	b	6	1	1	1	1	
2	a	3	1	1	-	2	1
	b	4	1	1	-	1	
	c	3	1	1	1	1	
3	a	9	1	1	-	3	1
	b	11	1	1	-	2	
4	a	5	1	1	-	1	3
5	a	9	2	1	-	1	4
	b	9	1	1	1	2	
	c	11	1	1	-	1	
6	a	4	1	1	-	2	5
	b	6	1	1	-	1	
7	a	8	1	-	-	1	5
8	a	7	1	1	-	1	6, 7
	b	9	1	1	-	-	8
9	a	5	1	1	1	1	8
	b	7	1	1	-	1	
10	a	1	1	1	-	-	9
	b	3	1	-	-	1	
11	a	3	1	1	-	1	9
	b	5	1	-	-	1	
12	a	1	1	-	1	-	9
	b	1	1	-	-	2	
13	a	5	-	1	1	2	9
	b	6	1	1	1	1	
	c	8	-	1	1	1	
14	a	3	-	1	1	2	7
	b	5	-	1	1	1	
15	a	5	1	1	1	-	10, 11, 12
	b	6	-	1	1	1	
16	a	3	1	1	1	-	15
	b	3	1	1	-	2	
17	a	13	2	1	1	1	13, 16
	b	15	1	1	1	1	
	c	13	1	2	1	2	
18	a	5	1	1	-	1	13, 15
	b	7	1	-	-	1	
19	a	2	1	1	-	1	14, 18
20	a	3	1	2	-	1	17
	b	5	1	1	-	1	
21	a	7	1	-	1	3	2, 4
	b	9	1	-	1	2	
	c	7	1	1	1	2	

PROBLEMA 3

FONTE ORIGINAL[1]: Feendley

RECURSO	R1	R2	R3	R4
DISPONIBILIDADE	*8*	*4*	*1*	*5*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS				ANTECESSORAS
			R1	R2	R3	R4	
1	a	25	6	1	-	2	-
	b	28	4	1	-	2	
	c	30	3	1	-	2	

2	a	40	5	-	1	2	1
	b	42	4	-	1	2	
	c	45	5	-	-	2	
3	a	37	1	3	1	4	2
	b	39	1	2	1	3	
4	a	15	4	1	1	2	3
	b	18	4	1	-	2	
5	a	20	7	2	1	2	1
	b	23	5	2	1	2	
	c	26	5	2	-	2	
6	a	21	8	1	1	-	2
	b	24	6	1	1	-	
	c	27	6	1	-	-	
7	a	52	2	1	1	1	3
	b	56	2	1	-	1	
8	a	48	3	3	-	4	5
	b	50	3	3	-	3	
	c	51	3	2	-	3	
9	a	17	5	-	1	3	5
	b	20	5	-	-	3	
10	a	17	4	1	1	4	1, 6
11	a	19	3	2	-	1	7
12	a	28	5	4	1	2	8
	b	30	5	2	1	2	
	c	31	5	4	-	2	
13	a	30	7	1	1	3	9
	b	32	6	1	1	3	
	c	35	7	1	-	3	
14	a	12	2	2	1	4	4, 10, 11
	b	15	2	2	-	4	
	c	13	2	2	1	2	
15	a	53	6	3	1	-	12, 13. 14

PROBLEMA 4

FONTE ORIGINAL[1]: Feendley

RECURSO	**R1**	**R2**
DISPONIBILIDADE	*1*	*1*

ATIVIDADE	**MODO**[2]	**DURAÇÃO**	**DEMANDA DE RECURSOS**		**ANTECESSORAS**
			R1	**R2**	
13	-	0	-	-	-
1	a	3	1	1	13
	b	5	1	-	
	c	6	-	1	
2	a	1	1	-	13
3	a	2	1	1	13
	b	5	1	-	
4	a	4	1	-	1
5	a	2	-	1	2
6	a	3	-	1	3
7	a	5	1	1	3, 6
	b	7	1	-	
	c	8	-	1	
8	a	3	-	1	5, 7
9	a	2	1	1	4
	b	3	1	-	
	c	5	-	1	
10	a	1	1	-	9
11	a	4	1	1	8, 9
	b	6	1	-	
	c	7	-	1	
12	a	6	1	-	10, 11

PROBLEMA 5

FONTE ORIGINAL[1]: Badiru

RECURSO	R1	R2	R3
DISPONIBILIDADE	*3*	*2*	*4*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
1	a	20	3	2	3	-
	b	24	2	1	3	
	c	22	2	2	3	
2	a	10	1	1	2	1
	b	12	1	1	1	
3	a	5	1	1	1	1
	b	7	1	-	1	
4	a	13	2	-	2	2
	b	15	1	-	2	
	c	17	1	-	1	
5	a	41	1	1	2	3
	b	45	1	-	2	
6	a	60	3	1	1	4
	b	63	2	1	1	
	c	67	1	1	1	
7	a	5	1	1	1	5
	b	8	1	-	1	
8	a	7	2	1	1	7
	b	8	1	1	1	
	c	10	1	-	1	
9	a	10	2	1	1	2
	b	14	1	-	1	
10	a	3	1	2	3	6
	b	5	1	1	3	
	c	6	1	1	2	
11	a	15	3	2	-	8, 9
	b	17	2	2	-	
	c	20	2	1	-	
12	a	2	1	-	1	10
13	a	4	1	1	1	11, 12
	b	6	1	-	1	
14	a	12	1	1	2	11
	b	14	1	-	2	
15	a	11	1	1	1	13
	b	14	1	-	1	
16	a	2	1	2	4	14, 15

PROBLEMA 6

FONTE ORIGINAL[1]: Wiest e Levy

RECURSO	R1	R2	R3	R4	R5
DISPONIBILIDADE	*5*	*7*	*4*	*2*	*3*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS					ANTECESSORAS
			R1	R2	R3	R4	R5	
1	a	1	4	4	2	1	2	-
	b	2	3	3	2	1	2	
	c	3	3	3	1	1	1	
2	a	2	2	6	1	1	3	1
	b	3	2	4	1	1	2	
	c	4	1	4	1	1	1	
3	a	2	1	2	1	1	1	2
	b	3	1	1	1	1	1	
	c	5	1	1	1	-	1	
4	a	1	3	3	4	2	-	3
	b	2	3	3	3	1	-	

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS					ANTECESSORAS
			R1	R2	R3	R4	R5	
	c	3	3	2	3	1	-	
5	a	5	2	4	2	2	1	4
	b	6	2	4	2	1	1	
	c	7	2	2	2	1	1	
6	a	1	1	2	3	1	1	5
	b	2	1	2	1	1	1	
	c	4	1	2	1	-	-	
7	a	3	1	3	2	1	2	5
	b	4	1	3	1	1	1	
	c	5	1	2	1	1	1	
8	a	2	-	1	-	1	2	5
	b	3	-	1	-	1	1	
9	a	1	3	-	1	1	1	6, 7, 8, 27
	b	3	2	-	1	-	1	9
10	a	3	4	7	1	1	-	
	b	4	4	6	1	1	-	
	c	6	3	5	1	1	-	
11	a	5	5	1	2	2	1	5
	b	7	4	1	2	1	1	
	c	8	3	1	2	1	1	
12	a	10	1	2	1	-	2	10, 11
	b	11	1	2	1	-	1	
13	a	3	1	3	3	1	1	12
	b	4	1	3	2	1	1	
	c	5	1	3	2	-	1	
14	a	1	2	5	1	1	3	13
	b	2	2	5	1	1	2	
	c	3	2	4	1	1	2	
15	a	1	3	2	1	1	2	10, 11
	b	2	2	2	1	1	2	
	c	3	2	2	1	1	1	
16	a	2	1	1	4	2	1	15
	b	3	1	1	3	2	1	
	c	4	1	1	3	1	1	
17	a	1	2	1	3	1	1	14, 16
	b	2	2	1	2	1	1	
	c	3	2	1	2	-	1	
18	a	5	4	1	2	-	2	17
	b	6	3	1	2	-	2	
	c	7	3	1	2	-	1	
19	a	2	1	2	1	1	3	18
	b	3	1	2	1	1	2	
	c	5	1	2	1	1	1	
20	a	2	1	1	3	1	2	18
	b	3	1	1	2	1	2	
	c	4	1	1	2	1	1	
21	a	1	-	4	-	1	2	18
	b	2	-	3	-	1	2	
	c	3	-	3	-	1	1	
22	a	1	4	5	1	1	1	7
	b	2	4	4	1	1	1	
	c	3	3	4	1	1	1	
23	a	3	1	-	1	2	1	5
	b	4	1	-	1	1	1	
	c	5	1	-	1	1	-	
24	a	1	2	3	2	-	2	8, 23
	b	2	2	2	2	-	2	
25	a	3	3	1	3	1	1	6
	b	4	3	1	2	1	1	
	c	5	2	1	2	1	1	
26	a	5	1	4	2	1	2	25
	b	6	1	3	2	1	2	
	c	7	1	3	2	1	1	
27	a	2	1	5	1	1	1	5
	b	3	1	4	1	1	1	
28	a	2	1	6	2	1	1	26, 27
	b	3	1	5	2	1	1	
	c	4	1	4	2	1	1	
29	a	2	3	1	4	-	2	28
	b	3	3	1	3	-	2	
	c	4	3	1	3	-	1	

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS					ANTECESSORAS
			R1	R2	R3	R4	R5	
30	a	1	2	3	-	1	3	10, 26
	b	2	2	3	-	1	2	
	c	3	2	3	-	-	2	
31	a	1	1	4	1	1	2	26
	b	2	1	4	1	1	1	
	c	3	1	3	1	1	1	
32	a	1	1	2	1	1	2	31
	b	2	1	2	1	1	1	
	c	4	1	2	1	1	-	
33	a	1	2	1	2	1	3	32, 37
	b	2	2	1	2	1	2	
	c	4	2	1	2	1	1	
34	a	2	3	2	-	2	1	33
	b	3	3	2	-	1	1	
	c	4	2	2	-	1	1	
35	a	1	1	4	2	1	2	20, 21, 22, 24
	b	2	1	3	2	1	2	
	c	4	1	3	2	1	1	
36	a	1	2	2	2	1	-	35
	b	2	2	2	1	1	-	
	c	4	2	2	1	-	-	
37	a	1	3	1	3	1	1	19, 29, 30, 35
	b	2	3	1	2	1	1	
	c	3	2	1	2	1	1	
38	a	2	4	3	-	1	1	37
	b	3	3	3	-	1	1	
	c	4	3	2	-	1	1	
39	a	1	1	2	1	1	3	18
	b	2	1	2	1	1	2	
	c	4	1	2	1	1	1	
40	a	1	5	4	3	1	2	34, 36, 38, 39
	b	2	4	4	3	1	2	
	c	5	4	4	2	1	1	

PROBLEMA 7

FONTE ORIGINAL[1]: Elmaghraby

RECURSO	R1	R2	R3	R4
DISPONIBILIDADE	*9*	*10*	*12*	*9*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS				ANTECESSORAS
			R1	R2	R3	R4	
60	-	-	-	-	-	-	-
1	a	4	-	-	2	-	60
	b	6	-	-	1	-	
2	a	3	8	-	-	-	1
	b	6	6	-	-	-	
3	a	2	5	-	-	-	60
	b	5	4	-	-	-	
4	a	2	-	-	8	-	3
	b	3	-	-	7	-	
5	a	2	-	-	-	3	4
6	a	1	-	2	-	-	60
7	a	1	-	-	7	-	6
	b	3	-	-	6	-	
8	a	2	-	-	-	5	7, 55
	b	4	-	-	-	4	
9	a	3	-	-	8	-	7 55
	b	5	-	-	6	-	
10	a	5	-	8	-	-	8
	b	8	-	6	-	-	
11	a	3	4	-	-	-	9
	b	4	3	-	-	-	
12	a	1	4	-	-	-	10

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS				ANTECESSORAS
			R1	R2	R3	R4	
	b	2	3	-	-	-	
13	a	4	-	-	9	-	11
14	a	1	-	5	-	-	12, 13
15	a	3	-	6	-	-	14
	b	5	-	5	-	-	
16	a	4	-	-	-	3	60
17	a	3	4	-	-	-	16
	b	6	3	-	-	-	
18	a	2	-	4	-	-	17
	b	3	-	3	-	-	
19	a	2	-	-	6	-	16
	b	4	-	-	5	-	
20	a	5	-	2	-	-	19
	b	8	-	1	-	-	
21	a	1	-	-	9	-	20
	b	5	-	-	7	-	
22	a	5	-	-	-	8	18
	b	9	-	-	-	6	
23	a	4	-	4	-	-	22
24	a	1	-	-	2	-	23
25	a	2	-	-	2	-	24
26	a	2	-	-	1	-	60
27	a	1	1	-	-	-	26
28	a	2	-	-	7	-	27
	b	5	-	-	5	-	
29	a	3	-	-	3	-	28, 31
	b	4	-	-	2	-	
	c	7	-	-	1	-	
30	a	4	-	1	-	-	26
31	a	2	-	-	-	4	30
	b	5	-	-	-	3	
32	a	1	-	7	-	-	30
	b	2	-	6	-	-	
	c	4	-	5	-	-	
33	a	2	3	-	-	-	32
	b	4	2	-	-	-	
34	a	3	-	-	-	8	29, 33
	b	4	-	-	-	7	
	c	7	-	-	-	6	
35	a	3	-	-	-	8	34
	b	5	-	-	-	7	
	c	7	-	-	-	6	
36	a	5	-	-	-	4	35
	b	7	-	-	-	3	
37	a	4	-	-	-	2	36
38	a	3	7	-	-	-	60
	b	4	6	-	-	-	
39	a	4	-	-	3	-	38
40	a	2	4	-	-	-	38
	b	4	3	-	-	-	
41	a	3	-	-	-	2	39, 53
	b	5	-	-	-	1	
42	a	5	-	6	-	-	41, 54
	b	7	-	5	-	-	
43	a	4	-	4	-	-	40
	b	6	-	3	-	-	
44	a	1	3	-	-	-	40
	b	4	2	-	-	-	
45	a	1	-	-	1	-	44
46	a	3	-	2	-	-	42, 43
	b	5	-	1	-	-	
47	a	3	-	-	2	-	45
48	a	2	9	-	-	-	45
	b	3	8	-	-	-	
	c	5	7	-	-	-	
49	a	5	-	-	-	8	48
	b	6	-	-	-	7	
	c	8	-	-	-	6-	
50	a	4	-	-	9	-	46, 47
	b	6	-	-	8	-	
51	a	1	3	-	-	-	49, 50

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS				ANTECESSORAS
			R1	R2	R3	R4	
	b	4	2	-	-	-	
52	a	10	8	-	-	-	60
	b	12	7	-	-	-	
	c	15	6	-	-	-	
53	a	1		-	-	4	26
	b	3	-	-	-	3	
54	a	2	-	-	-	5	38
	b	3	-	-	-	4	
	c	6	-	-	-	3	
55	a	4	-	-	-	1	3
70	a	-	-	-	-	-	52, 2, 5, 15, 25, 21, 53, 37, 51

PROBLEMA 8

FONTE ORIGINAL[1]: Elmaghraby

RECURSO	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
DISPONIBILIDADE	*12*	*4*	*7*	*15*	*2*	*5*	*6*	*10*	*3*	*1*

ATIVIDADE	**MODO**[2]	**DURAÇÃO**	**DEMANDA DE RECURSOS**										**ANTECESSORAS**
			R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	
1	a	4	-	3	5	-	1	-	3	-	-	-	-
2	a	2	7	2	2	-	-	2	-	5	2	1	1
	b	3	6	2	2	-	-	2	-	5	1	1	
3	a	3	5	1	-	3	-	3	-	2	-	-	2
	b	5	4	1	-	3	-	2	-	2	-	-	
4	a	6	4	3	-	4	-	1	1	-	-	1	2
	b	9	4	3	-	4	-	1	1	-	-	-	
5	a	5	1	2	-	-	-	-	2	-	-	-	2
6	a	1	-	-	4	-	1	-	3	-	-	-	1
	b	3	-	-	4	-	1	-	1	-	-	-	
7	a	2	2	-	-	1	-	-	-	-	1	-	6
	b	4	2	-	-	1	-	-	-	-	-	-	
8	a	3	3	-	-	-	1	-	-	2	-	-	6
	b	4	7	-	-	-	1	-	-	-	-	-	
9	a	7	5	-	-	-	1	-	-	-	-	-	3, 8
10	a	3	9	-	-	-	2	1	-	-	-	-	2, 6, 8
	b	5	8	-	-	-	1	1	-	-	-	-	
11	a	1	11	1	-	-	-	-	-	4	-	-	10
	b	4	8	1	-	-	-	-	-	-	-	-	
12	a	4	5	-	1	2	1	-	4	-	-	-	9
	b	5	5	-	1	2	1	-	3	-	-	-	
13	a	4	4	-	-	1	1	3	-	-	-	-	4, 9
	b	6	4	-	-	1	-	3	-	-	-	-	
14	a	7	3	-	1	8	-	2	-	-	-	-	3
	b	8	3	-	1	6	-	2	-	-	-	-	
15	a	8	2	2	-	3	-	1	-	-	-	-	5
16	a	6	5	1	-	2	-	4	1	-	-	-	3
	b	8	4	1	-	2	-	3	1	-	-	-	
17	a	7	-	2	-	1	-	-	4	-	-	-	2
18	a	3	-	3	-	6	-	1	-	-	-	-	5
	b	5	-	2	-	4	-	1	-	-	-	-	
19	a	2	-	2	1	-	1	-	-	8	1	-	12, 13
	b	5	-	2	1	-	1	-	-	6	1	-	
20	a	4	1	1	2	-	-	-	-	-	-	-	14, 15

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS										ANTECESSORAS
			R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	
21	a	12	5	-	-	2	-	1	-	-	-	-	15
	b	15	3	-	-	2	-	1	-	-	-	-	
22	a	5	-	-	-	12	-	2	-	-	-	-	16
	b	8	-	-	-	9	-	2	-	-	-	-	
23	a	9	-	2	-	-	1	3	-	-	-	-	6
	b	10	-	2	-	-	1	2	-	-	-	-	
24	a	6	1	-	-	-	-	-	-	-	2	1	8
	b	8	1	-	-	-	-	-	-	-	1	1	
25	a	9	-	-	-	9	-	1	-	-	-	-	6
	b	11	-	-	-	7	-	1	-	-	-	-	
26	a	8	4	-	2	2	-	-	-	-	-	-	7
27	a	9	5	-	-	1	-	1	-	-	-	-	7
28	a	9	3	-	-	1	-	2	4	-	-	-	16, 17, 23
	b	12	3	-	-	1	-	1	3	-	-	-	
29	a	8	4	-	5	-	1	3	-	-	-	-	18
	b	10	4	-	4	-	1	2	-	-	-	-	
30	a	6	-	3	-	-	-	1	-	-	-	-	24, 25
31	a	8	1	1	-	3	-	-	-	-	-	1	19, 20, 21
32	a	4	-	-	-	7	-	-	1	-	-	-	21
	b	5	-	-	-	5	-	-	1	-	-	-	
33	a	13	2	-	-	1	1	1	-	-	-	-	22
34	a	10	2	-	-	-	-	-	-	9	-	-	31, 32
	b	13	2	-	-	-	-	-	-	7	-	-	
35	a	3	-	-	-	2	-	-	-	-	-	-	32
	b	6	-	-	-	2	-	-	-	-	3	-	
36	a	13	3	-	1	3	1	-	-	-	1	-	28, 33
	b	16	3	-	1	3	-	-	-	-	-	-	
37	a	7	-	2	-	-	-	2	-	-	-	-	34, 35
38	a	8	-	-	-	5	-	3	-	-	-	-	26, 27
	b	9	-	-	-	5	-	2	-	-	-	-	
39	a	12	-	-	-	1	-	-	2	-	1	-	25
40	a	5	-	3	4	-	-	1	-	-	-	-	28, 39
41	a	4	-	2	-	-	-	2	-	-	-	-	39
42	a	8	10	-	-	-	1	3	-	-	-	-	36
	b	12	8	-	-	-	1	2	-	-	-	-	
43	a	8	3	-	-	-	-	-	5	2	-	-	41
	b	11	3	-	-	-	-	-	3	2	-	-	
44	a	11	1	-	-	2	-	1	-	1	-	-	40, 41
45	a	6	2	1	-	-	1	2	-	2	-	-	43
46	a	6	-	-	4	-	2	-	-	-	-	-	34, 35, 43
	b	8	-	-	4	-	1	-	-	-	-	-	
47	a	3	1	-	2	1	-	-	-	4	-	-	45, 46

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS										ANTECESSORAS
			R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	
48	a	7	2	-	-	1	-	1	-	-	-	-	44
49	a	14	4	-	-	-	-	1	-	-	2	-	10
	b	15	4	-	-	-	-	1	-	-	1	-	
50	a	3	-	3	-	6	-	2	-	-	-	-	11
51	a	7	5	-	-	7	-	3	-	-	-	-	11
	b	9	5	-	-	6	-	2	-	-	-	-	
52	a	10	-	1	2	-	-	-	-	5	-	-	50
53	a	9	-	-	5	2	-	1	-	2	-	-	52
	b	12	-	-	3	2	-	1	-	2	-	-	
54	a	6	5	1	2	6	1	-	-	-	-	-	52
	b	8	4	1	2	5	1	-	-	-	-	-	
55	a	10	1	2	-	7	-	-	-	-	-	-	51, 52
56	a	3	2	-	-	-	1	-	-	3	-	-	52
57	a	6	3	-	1	-	-	-	3	-	-	1	53
58	a	13	4	-	-	8	-	1	-	-	-	-	30, 38, 53
	b	15	4	-	-	6	-	1	-	-	-	-	
59	a	6	5	-	-	2	1	2	-	-	-	-	54, 55
	b	9	4	-	-	2	-	2	-	-	-	-	
60	a	10	-	1	-	5	-	-	-	-	1	-	56
61	a	4	-	2	-	2	-	-	-	3	-	-	57
62	a	3	-	3	5	-	-	-	-	-	-	-	57
	b	5	-	2	4	-	-	-	-	-	-	-	
63	a	3	-	-	-	12	1	-	2	-	-	-	61
	b	6	-	-	-	9	1	-	2	-	-	-	
64	a	8	-	-	-	8	-	1	-	-	2	-	61
	b	10	-	-	-	7	-	1	-	-	1	-	
65	a	8	1	-	-	5	-	1	-	-	-	-	62
66	a	12	-	2	-	3	-	2	-	-	-	-	58, 62
67	a	7	-	1	3	-	-	-	-	-	2	-	49, 59, 60, 61
68	a	6	5	1	-	-	1	-	-	-	-	-	49, 59, 60, 61
69	a	7	4	1	-	-	2	-	-	-	-	-	66
	b	9	4	1	-	-	1	-	-	-	-	-	
70	a	13	-	-	1	7	-	-	-	-	1	-	47, 63, 64, 67, 68
71	a	5	-	-	2	-	1	-	-	4	-	-	37, 42, 48, 65, 69
72	a	5	2	1	-	-	1	-	-	-	-	-	37, 42, 48, 65, 69
	b	7	2	1	-	-	-	-	-	-	-	-	
73	a	4	7	3	-	14	-	-	-	9	-	1	70, 71, 72
74	a	-	-	-	-	-	-	-	-	-	-	-	29, 73

PROBLEMA 9

FONTE ORIGINAL[1]: Davis e Patterson (modificado por Badiru. Os valores originalmente proposto por estes autores são mostrados no problema 25)

RECURSO	R1	R2	R3
DISPONIBILIDADE	6	6	6

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
1	-	1	-	-	-	-
2	a	2	3	5	2	1
	b	6	3	3	1	
	c	7	2	4	1	
3	a	5	5	4	3	1
	b	6	4	4	3	
	c	8	3	3	3	
4	a	3	5	2	2	1
	b	4	4	2	2	
	c	6	3	2	2	
5	a	4	4	1	4	2, 3
	b	5	3	1	4	
	c	6	3	2	2	
6	a	1	5	5	4	4
	b	3	4	4	4	
	c	5	3	3	3	
7	a	6	3	5	2	4
	b	7	3	4	2	
	c	8	3	3	3	
8	a	8	2	4	4	2, 3
	b	3	2	3	4	
	c	4	2	3	3	
9	a	6	3	2	2	5, 7, 8
	b	7	2	2	2	
	c	8	2	1	1	
10	a	3	3	2	4	6, 7, 8
	b	4	2	2	4	
	c	6	2	2	2	
11	a	3	3	3	2	6
	b	4	2	3	2	
	c	5	2	2	2	
12	a	1	4	1	4	5
	b	2	3	1	4	
	c	3	3	1	3	
13	a	3	1	4	4	9, 10
	b	4	1	3	3	
	c	6	1	2	2	
14	a	6	2	2	2	12
	b	7	1	2	2	
15	a	3	5	5	4	9
	b	5	4	4	4	
	c	7	3	4	3	
16	a	4	1	5	4	11
	b	5	1	4	4	
	c	6	1	4	3	
17	a	4	4	5	4	15, 16
	b	6	3	4	4	
	c	8	3	3	3	
18	a	3	3	2	3	14
	b	4	2	2	3	
19	a	3	5	3	3	13
	b	4	4	3	3	
	c	6	3	3	3	
20	a	1	2	4	6	16
	b	2	2	4	5	
	c	4	2	4	4	
21	a	4	1	6	2	18, 19

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
	b	5	1	5	2	
	c	7	1	4	2	
22	a	6	3	2	1	17, 20
	b	7	2	2	1	
23	a	4	1	-	4	19
	b	5	1	-	3	
24	a	1	2	2	1	17
	b	2	1	2	1	
25	a	3	-	1	3	21, 22, 23, 24
	b	4	-	1	2	
26	a	3	2	2	2	23
	b	4	1	2	2	
27	a	1	-	-	-	25, 26

PROBLEMA 10

FONTE ORIGINAL[1]: Davis

RECURSO	R1	R2
DISPONIBILIDADE	*3*	*7*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS		ANTECESSORAS
			R1	R2	
1	a	4	-	-	-
2	a	2	1	-	1
	b	3	-	2	
3	a	3	1	-	2
	b	4	-	2	
4	a	3	-	-	2
5	a	5	1	-	2
	b	6	-	2	
6	a	4	1	-	3, 29
	b	5	-	2	
7	a	4	1	-	6
	b	5	-	3	
8	a	4	-	1	4, 6
9	a	7	-	1	3
10	a	8	1	-	5
	b	9	-	2	
11	a	6	1	-	3
	b	7	-	2	
12	a	7	-	1	2
13	a	3	1	-	5
	b	5	-	1	
14	a	2	-	1	7, 8
15	a	4	-	1	9, 10
16	a	12	1	-	10
	b	13	-	2	
17	a	5	1	-	11
	b	7	-	1	
18	a	9	-	2	11, 12, 30
	b	11	-	1	
19	a	8	-	1	13
20	a	8	-	2	14, 15, 16
	b	10	-	1	
21	a	4	1	-	16
	b	5	-	2	
22	a	13	1	-	17
	b	15	-	1	

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS		ANTECESSORAS
			R1	R2	
23	a	10	-	1	20, 21
24	a	3	-	1	21
25	a	13	-	1	18, 22
26	a	7	-	1	23, 24
27	a	8	-	1	25
28	a	1	1	-	1
	b	3	-	1	
29	a	3	1	-	2, 28
	b	4	-	2	
30	a	9	-	1	28
31	a	6	-	1	29
32	a	9	1	-	28
33	a	2	1	-	28
	b	3	-	2	
34	a	8	1	-	33
	b	10	-	1	
35	a	9	-	1	33
36	a	6	-	1	31, 32
37	a	12	1	-	32
38	a	8	-	1	34, 35
39	a	4	1	-	37
40	a	5	-	1	18, 37
41	a	8	-	1	39
42	a	11	-	1	39, 40
43	a	6	-	3	24, 41
	b	7	1	1	
44	a	6	1	-	41
	b	8	-	1	
45	a	7	-	1	42
46	a	3	-	3	43, 44
	b	4	-	2	
47	a	3	1	-	2, 28, 29
48	a	1	1	-	47
49	a	14	-	1	47
50	a	3	1	-	48
51	a	7	-	1	48
52	a	10	2	-	50
	b	12	1	1	
53	a	9	2	-	52
	b	11	-	2	
54	a	6	1	-	52
	b	8	-	1	
55	a	10	-	2	51, 52
	b	12	-	1	
56	a	3	2	-	52
	b	3	-	3	
57	a	6	2	-	53
	b	7	-	2	
58	a	13	-	2	36, 38, 53
	b	13	1	1	
59	a	6	-	2	54, 55
	b	8	-	1	
60	a	10	-	1	56
61	a	4	1	-	57
	b	4	-	2	
62	a	3	1	-	57
	b	3	-	2	
63	a	3	1	-	61
64	a	8	1	-	61
65	a	8	1	-	62
66	a	12	-	3	58, 62
	b	12	1	1	
67	a	7	-	1	49, 59, 60
68	a	6	-	1	49, 59, 60, 61

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS		ANTECESSORAS
			R1	R2	
69	a	7	-	3	66
	b	7	1	2	
70	a	13	-	-	46, 63, 64, 67, 68
71	a	5	-	3	26, 27, 45, 65, 69
	b	7	-	2	
72	a	5	-	3	26, 27, 45, 65, 69
	b	5	1	1	
73	a	4	-	-	70, 71, 72
74	a	3	-	1	26
75	a	1	1	1	68, 69
	b	3	1	-	
76	a	1	1	-	27, 74
77	a	4	2	3	46, 76
	b	6	1	2	
78	a	2	1	1	69, 75
79	a	3	1	1	78
80	a	9	2	2	73, 77, 79
	b	10	1	2	

PROBLEMA 11

FONTE ORIGINAL[1]: Dar-El e Tur

RECURSO	R1	R2
DISPONIBILIDADE	*2*	*2*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS		ANTECESSORAS
			R1	R2	
51	a	10	-	-	-
1	a	25	-	-	51
2	a	5	1	-	51
	b	6	-	1	
3	a	2	-	2	51
	b	4	1	-	
4	a	3	1	1	51
	b	5	1	-	
5	a	4	-	1	51
	b	4	1	-	
6	a	7	-	1	5
7	a	12	-	1	1
8	a	15	-	1	1
9	a	2	1	1	1, 2
	b	4	-	1	
10	a	5	-	1	2, 3
11	a	4	2	-	
	b	6	1	-	
	c	5	1	1	
12	a	2	1	1	4, 7, 9
	b	3	1	-	
	c	4	-	1	
13	a	5	1	1	8, 11
	b	7	1	-	
14	a	6	1	1	4, 6
	b	7	-	1	
15	a	11	-	1	14
16	a	12	1	1	10, 15
	b	13	1	-	
	b	14	-	1	

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS		ANTECESSORAS
			R1	R2	
17	a	8	1	-	6
18	a	19	1	-	6
19	a	20	1	1	6
20	a	7	-	1	51
21	a	7	1	2	20
	b	9	1	1	
	c	8	2	-	
22	a	8	1	-	19, 21
23	a	2	-	1	17
24	a	3	-	1	23
25	a	4	-	2	16, 18
	b	5	1	1	
	c	6	2	-	
26	a	8	1	-	25
	b	9	-	1	
27	a	11	2	-	12, 13
	b	12	1	1	
	c	13	1	-	
28	a	10	1	-	26
29	a	10	1	-	28
30	a	11	-	2	28
	b	12	1	1	
	c	13	-	1	
31	a	8	1	-	28
32	a	4	-	2	30, 31
	b	6	-	1	
33	a	3	-	1	32
34	a	2	-	1	24
35	a	8	1	-	24
	b	9	-	1	
36	a	7	1	2	26
	b	9	1	1	
37	a	7	1	-	35, 36
	b	9	-	1	
38	a	20	1	-	34
39	a	8	1	-	34
40	a	19	1	1	34
	b	21	1	-	
	c	20	-	1	
41	a	5	-	1	40
	b	6	1	-	
42	a	2	-	1	41
	b	3	1	-	
43	a	3	2	-	42
	b	5	1	-	
	c	4	-	2	
44	a	5	2	-	39
	b	7	1	-	
45	a	3	1	-	44
46	a	12	-	1	45
47	a	15	-	2	38
	b	17	-	1	
48	a	7	-	-	47
	b	8	1	1	
49	a	4	1	-	48
	b	5	-	1	
50	a	3	1	1	27, 29, 33, 43, 49
	b	5	1	-	
	c	6	-	1	
52	a	-	-	-	22, 37, 46, 50

PROBLEMA 12

FONTE ORIGINAL[1]: Talbot

RECURSO	R1	R2	R3	R4
DISPONIBILIDADE	*1*	*2*	*6*	*8*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS				ANTECESSORAS
			R1	R2	R3	R4	
10	-	-	-	-	-	-	-
1	a	2	1	-	2	1	10
	b	4	-	-	2	1	
2	a	1	1	-	3	2	10
	b	5	-	-	2	2	
3	a	3	1	-	1	4	10
	b	6	-	-	1	4	
	c	5	1	-	1	2	
4	a	5	1	-	1	3	1
	b	6	1	-	-	3	
	c	8	-	-	1	3	
5	a	4	1	-	1	2	1
	b	5	1	-	1	1	
	c	6	-	-	2	2	
6	a	1	1	-	3	4	2, 4
	b	3	1	-	2	2	
	c	4	-	-	3	4	
7	a	1	-	-	-	-	3, 5, 6

PROBLEMA 13

FONTE ORIGINAL[1]: Davis e Heidorn

RECURSO	R1	R2	R3
DISPONIBILIDADE	*5*	*5*	*3*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
1	-	1	-	-	-	-
2	a	1	2	2	1	1
	b	3	2	1	1	
	c	4	1	2	1	
3	a	2	-	2	1	1
	b	4	-	1	1	
4	a	2	3	3	3	2
	b	4	3	3	2	
	c	6	2	2	2	
5	a	3	2	1	3	2
	b	5	2	1	2	
	c	7	2	1	1	
6	a	2	1	1	-	3, 5
7	a	1	-	-	-	4, 6

PROBLEMA 14

FONTE ORIGINAL[1]: Stinson, *et al.*

RECURSO	**R1**	**R2**	**R3**	**R4**
DISPONIBILIDADE	*10*	*9*	*9*	*12*

ATIVIDADE	**MODO**[2]	**DURAÇÃO**	**DEMANDA DE RECURSOS**				**ANTECESSORAS**
			R1	**R2**	**R3**	**R4**	
1	-	1	-	-	-	-	-
2	a	4	3	2	4	2	1
	b	5	2	2	4	2	
	c	7	2	2	3	2	
3	a	6	6	2	3	1	1
	b	7	5	2	3	1	
	c	8	4	2	3	1	
4	a	6	5	4	3	2	1
	b	8	4	3	3	2	
	c	10	4	3	2	2	
5	a	2	1	3	3	1	2, 3
	b	3	1	2	3	1	
	c	5	1	2	2	1	
6	a	2	4	1	6	2	3, 4
	b	3	4	1	5	2	
7	a	3	4	1	4	9	2
	b	5	4	1	4	7	
8	a	5	1	2	2	2	3, 4
9	a	7	1	5	2	3	6, 8
	b	6	1	4	2	3	
	c	8	1	4	2	2	
10	a	6	6	1	5	3	8
	b	8	5	1	4	3	
11	a	8	6	1	6	3	8
	b	10	5	1	5	3	
12	a	2	5	1	6	6	5, 6
	b	3	5	1	5	5	
	c	5	4	1	5	5	
13	a	1	3	4	4	3	9
14	a	7	1	1	1	1	12
15	a	2	1	1	4	1	11
	b	3	1	1	3	1	
	c	5	1	1	2	1	
16	a	1	1	1	2	3	10
	b	2	1	1	2	2	
	c	3	1	1	1	2	
17	a	9	1	3	5	2	14
	b	10	1	3	4	2	
	c	12	1	2	4	2	
18	a	5	4	2	3	1	13, 15
	b	6	3	2	3	1	
	c	8	3	2	2	1	
19	a	3	5	1	4	3	15, 16
	b	4	5	1	3	3	
20	a	3	2	2	5	3	14
	b	4	2	2	4	3	
21	a	4	3	5	6	5	18
	b	6	3	5	5	5	
22	a	5	2	6	6	1	18, 20
	b	6	2	5	6	1	
	c	8	2	5	5	1	
23	a	1	3	2	4	1	19
	b	2	3	2	3	1	
24	a	1	2	6	4	3	17
	b	2	2	5	4	3	
	c	5	2	4	3	3	
25	a	1	1	1	2	4	24

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS				ANTECESSORAS
			R1	R2	R3	R4	
26	a	5	6	3	6	1	23
	b	8	5	3	5	1	
27	a	6	6	2	2	6	22
	b	8	5	2	2	5	
28	a	3	5	1	5	1	21
	b	6	4	1	4	1	
29	a	3	3	3	2	4	25, 28
30	a	7	4	3	4	3	25, 26
31	a	5	2	5	3	3	26, 27
	b	7	2	4	3	3	
32	a	2	1	3	5	5	26
33	a	3	2	3	3	2	30
34	a	6	4	1	4	1	32
35	a	7	3	5	5	4	29
36	a	1	1	4	5	4	31, 32
	b	3	1	4	4	4	
37	a	3	1	5	1	2	34
	b	4	1	4	1	2	
38	a	2	5	3	1	3	33, 35
39	a	2	2	4	3	3	35
	b	4	2	2	3	3	
40	a	1	6	3	2	1	35
	b	2	5	3	2	1	
	c	4	4	3	2	1	
41	a	2	2	6	5	4	39
	b	3	2	5	5	4	
	c	5	2	5	4	4	
42	a	3	3	3	2	4	37, 40
43	a	-	-	-	-	-	7, 36, 38, 41, 42

PROBLEMA 15

FONTE ORIGINAL[1]: Gorenstein

RECURSO	R1	R2	R3	R4
DISPONIBILIDADE	*3*	*3*	*3*	*2*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS				ANTECESSORAS
			R1	R2	R3	R4	
14	-	-	-	-	-	-	-
1	a	2	2	-	-	-	14
	b	4	1	-	-	-	
2	a	3	-	1	-	-	1
3	a	4	1	-	-	-	14
4	a	5	-	-	-	2	3
	b	7	-	-	-	1	
5	a	6	-	3	-	-	4
	b	7	-	2	-	-	
	c	9	-	1	-	-	
6	a	7	3	-	-	-	3, 9
	b	9	2	-	-	-	
7	a	3	-	2	-	-	6
	b	5	-	1	-	-	
8	a	4	-	-	1	-	7
9	a	2	1	-	-	-	14
10	a	1	-	-	2	-	9
	b	3	-	-	1	-	
11	a	10	-	-	-	1	10
12	a	8	-	-	1	-	14
13	a	4	-	-	3	1	12

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS				ANTECESSORAS
			R1	R2	R3	R4	
	b	6	-	-	2	1	
	c	8	-	-	1	1	
15	a	-	-	-	-	-	2, 5, 8, 11, 13

PROBLEMA 16

FONTE ORIGINAL[1]: Badiru

RECURSO	R1	R2
DISPONIBILIDADE	*2*	*3*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS		ANTECESSORAS
			R1	R2	
8	-	-	-	-	-
1	a	2	1	1	8
	b	4	1	-	
	c	6	-	1	
2	a	6	-	2	8
3	a	4	-	1	8
4	a	3	1	-	1
5	a	5	2	1	3
	b	7	1	1	
	c	7	1	2	
6	a	4	1	1	1
	b	7	1	-	
	c	8	-	1	
7	a	2	2	3	4, 5

PROBLEMA 17

FONTE ORIGINAL[1]: Willis

RECURSO	R1	R2	R3
DISPONIBILIDADE	*3*	*6*	*10*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
1	a	2	2	4	8	-
	b	4	1	3	7	
	c	5	1	2	6	
2	a	3	3	2	4	1
	b	5	2	2	4	
	c	7	1	2	4	
3	a	4	1	5	9	1
	b	6	1	4	8	
	c	7	1	4	7	
4	a	1	1	4	5	1
	b	3	1	3	5	
	c	5	-	3	5	
5	a	4	1	1	4	2
	b	5	1	1	3	
	c	7	-	1	3	
6	a	3	2	2	8	3

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
	b	4	1	2	8	
	c	6	1	2	6	
7	a	5	1	4	4	3
	b	7	1	3	3	
	c	10	1	2	2	
8	a	4	1	2	1	4, 5, 6
	b	6	-	2	1	
	c	8	-	1	1	
9	a	3	2	5	4	7, 8

PROBLEMA 18

FONTE ORIGINAL[1]: Willis

RECURSO	R1	R2	R3
DISPONIBILIDADE	*1*	*1*	*1*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
13	-	-	-	-	-	-
1	a	4	-	-	1	13
2	a	1	-	-	1	12
3	a	1	-	-	1	12
4	a	3	1	-	-	1
	b	5	-	1	-	
5	a	1	1	-	-	2
	b	3	-	1	-	
6	a	3	1	-	-	2
	b	4	-	1	-	
	c	5	-	-	1	
7	a	5	-	1	-	2
	b	6	-	-	1	
8	a	4	-	1	-	3
	b	6	-	-	1	
9	a	8	1	-	-	4, 5
	b	10	-	1	-	
	c	12	-	-	1	
10	a	6	-	1	-	6, 8
	b	8	-	-	1	
11	a	1	1	-	-	6, 8
	b	4	-	1	-	
12	a	1	-	1	-	7, 9, 10
14	a	-	-	-	-	11, 12

PROBLEMA 19

FONTE ORIGINAL[1]: Brand, Mayer e Schaffer

RECURSO	R1	R2	R3
DISPONIBILIDADE	*2*	*1*	*2*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
14		-	-	-	-	-
1		12	1	-	-	14
2		8	-	-	-	14
3		7	-	-	-	14
4		9	-	1	1	1
		10	-	1	-	
		12	-	-	1	
5		5	1	-	-	2
6		3	-	-	-	2
7		12	1	-	1	2
		14	-	-	1	
		13	1	-	-	
8		4	-	-	1	3, 5
9		2	-	-	-	3, 5
10		7	-	-	1	6, 1
11		6	-	1	-	7, 8, 10
12		10	-	-	-	11, 9
15		-	-	-	-	4, 12

PROBLEMA 20

FONTE ORIGINAL[1]: Nemeti

RECURSO	R1	R2	R3	R4
DISPONIBILIDADE	*1*	*1*	*1*	*1*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS				ANTECESSORAS
			R1	R2	R3	R4	
14		-	-	-	-	-	-
1	a	2	1	-	-	-	14
	b	4	-	1	-	-	
2	a	3	1	-	-	-	14
3	a	1	1	-	-	-	14
4	a	4	1	-	-	-	14
	b	7	-	-	1	-	
5	a	4	-	-	-	1	14
6	a	3	-	1	-	-	1
7	a	3	-	-	1	-	2
8	a	3	-	1	-	-	3
	b	5	-	-	1	-	
9	a	1	-	-	-	1	4
10	a	4	-	-	1	-	5
	b	7	-	-	-	1	
11	a	2	-	1	-	-	7
	b	6	-	-	1	-	
12	a	2	-	-	-	1	8
	b	3	-	-	1	-	
13	a	3	-	-	1	-	9
15	a	-	-	-	-	-	6, 10, 11, 12, 13

PROBLEMA 21

FONTE ORIGINAL[1]: Willis

RECURSO	R1	R2
DISPONIBILIDADE	*3*	*4*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS		ANTECESSORAS
			R1	R2	
16	-	-	-	-	-
1	a	18	1	1	16
	b	20	1	-	
	c	22	-	1	
2	a	18	1	2	16
	b	20	1	1	
3	a	6	-	1	16
4	a	12	1	1	16
	b	13	1	-	
5	a	12	-	2	1
	b	14	-	1	
6	a	4	-	1	1
7	a	24	1	1	2
	b	25	1	-	
	c	27	-	1	
8	a	20	1	2	3
	b	23	1	1	
9	a	14	2	2	5, 6
	b	15	2	1	
	c	17	1	2	
10	a	20	2	1	4
	b	22	1	1	
1	a	18	1	1	8
12	a	14	1	1	8
13	a	30	1	1	2, 9
	b	32	1	-	
	c	33	-	1	
14	a	6	2	1	7, 10, 11, 13
	b	8	1	1	
17	a	-	-	-	12, 14

PROBLEMA 22

FONTE ORIGINAL[1]: Li e Willis

RECURSO	R1	R2	R3	R4	R5	R6
DISPONIBILIDADE	*4*	*2*	*1*	*3*	*1*	*1*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS						ANTECESSORAS
			R1	R2	R3	R4	R5	R6	
18	-	-	-	-	-	-	-	-	-
1	a	1	-	-	-	-	-	-	18
2	a	2	1	-	-	-	-	1	18
	b	5	1	-	-	-	-	-	
3	a	6	-	-	-	-	-	-	18
4	a	3	2	1	-	-	-	-	18
	b	5	1	1	-	-	-	-	
5	a	3	1	-	-	-	1	-	1
	b	5	1	-	-	-	-	-	
6	a	2	2	1	-	-	-	-	5

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS						ANTECESSORAS
			R1	R2	R3	R4	R5	R6	
	b	4	1	1	-	-	-	-	
7	a	9	4	-	-	-	-	-	5
	b	11	3	-	-	-	-	-	
	c	13	2	-	-	-	-	-	
8	a	10	4	-	-	-	-	-	2, 5
	b	12	3	-	-	-	-	-	
9	a	4	1	-	1	-	-	-	4, 3
10	a	14	2	2	-	-	-	-	4, 3
	b	16	2	1	-	-	-	-	
11	a	4	1	-	1	-	-	-	8
12	a	2	3	-	-	-	-	-	6, 11
	b	5	2	-	-	-	-	-	
13	a	3	1	-	-	1	-	-	10
14	a	1	-	-	-	-	-	-	12
15	a	4	2	2	-	-	-	-	14, 13
	b	6	2	1	-	-	-	-	
19	a	-	-	-	-	-	-	-	7, 15

PROBLEMA 23

FONTE ORIGINAL[1]: Davis

RECURSO	R1	R2	R3
DISPONIBILIDADE	*10*	*10*	*10*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
40	-	-	-	-	-	-
1	a	3	3	5	2	40
	b	4	3	3	2	
2	a	3	5	2	2	40
	b	6	3	2	2	
3	a	4	5	4	3	40
	b	5	4	4	3	
	c	6	4	3	3	
4	a	5	2	4	4	1
	b	8	2	3	3	
5	a	4	4	1	4	1, 2
	b	7	2	1	4	
6	a	6	3	5	2	3
	b	9	3	3	2	
7	a	3	5	5	4	3
	b	7	4	4	4	
8	a	4	3	3	2	4
	b	5	2	3	2	
	c	7	2	2	2	
9	a	2	3	2	2	4, 5
	b	3	2	2	2	
10	a	3	1	4	4	8, 9
	b	6	1	3	3	
11	a	8	4	1	4	6
	b	10	3	1	3	
12	a	2	3	2	4	6, 7
	b	3	2	2	4	
	c	5	2	2	3	
13	a	2	1	5	4	11
	b	5	1	3	4	
14	a	3	5	5	4	11
	b	6	4	4	4	
15	a	2	2	2	2	11, 12

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
16	a	2	3	2	3	13
	b	3	3	2	2	
17	a	3	2	4	6	14
	b	5	2	4	4	
18	a	3	4	5	4	10, 14
	b	5	4	4	4	
19	a	3	1	-	4	17
	b	5	1	-	3	
20	a	2	2	2	2	17
21	a	2	1	6	2	16, 17, 18
	b	3	1	5	2	
22	a	3	2	2	2	19
23	a	5	5	3	3	15
	b	8	3	3	3	
24	a	5	3	2	1	23
25	a	3	-	1	3	19, 20, 21
50	-	-	-	-	-	21, 22, 24, 25

PROBLEMA 24

FONTE ORIGINAL[1]: Willis e Hastings

RECURSO	R1	R2	R3	R4	R5	R6
DISPONIBILIDADE	*10*	*5*	*4*	*2*	*1*	*2*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS						ANTECESSORAS
			R1	R2	R3	R4	R5	R6	
40	-	-	-	-	-	-	-	-	-
1	a	5	2	1	-	-	-	-	40
	b	7	1	1	-	-	-	-	
2	a	8	6	-	-	-	-	1	40
	b	11	4	-	-	-	-	1	
3	a	4	4	-	-	-	-	1	1
	b	6	3	-	-	-	-	1	
4	a	3	2	-	2	-	-	-	3
	b	4	2	-	1	-	-	-	
	c	6	1	-	1	-	-	-	
5	a	4	3	1	-	-	1	-	4
	b	8	3	1	-	-	-	-	
6	a	4	3	-	-	-	-	1	4
	b	5	2	-	-	-	-	1	
7	a	1	-	2	-	-	-	-	5
8	a	3	-	2	-	-	-	-	5
9	a	1	-	2	-	-	-	-	7, 8
10	a	3	2	-	-	2	-	-	8
	b	5	2	-	-	1	-	-	
11	a	6	2	1	-	2	-	-	8
	b	8	2	1	-	1	-	-	
	c	9	1	1	-	1	-	-	
12	a	1	1	1	-	-	-	-	9, 10
13	a	4	4	-	-	-	-	1	2, 3
	b	5	3	-	-	-	-	1	
14	a	4	1	-	-	-	-	-	13
15	a	12	2	-	4	-	-	-	13
	b	14	2	-	3	-	-	-	
16	a	6	1	-	-	-	-	1	13
17	a	1	2	1	-	-	1	-	5, 14
18	a	1	2	2	-	-	-	-	17
19	a	1	2	2	-	-	-	-	15
	b	3	2	1	-	-	-	-	

ATIIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS						ANTECESSORAS
			R1	R2	R3	R4	R5	R6	
20	a	1	2	2	-	-	-	-	18, 19
	b	2	1	2	-	-	-	-	
21	a	2	1	-	2	-	-	-	15, 16
22	a	4	1	-	-	2	-	-	21
	b	6	1	-	-	1	-	-	
23	a	1	2	2	-	-	1	-	22
	b	4	2	2	-	-	-	-	
24	a	9	4	-	-	-	-	1	6
25	a	9	4	-	-	-	-	-	6
	b	11	3	-	-	-	-	-	
26	a	4	2	2	-	-	-	-	24
	b	5	2	1	-	-	-	-	
	c	6	1	1	-	-	-	-	
27	a	6	2	-	2	-	-	-	21, 24
	b	8	2	-	1	-	-	-	
	c	10	1	-	1	-	-	-	
28	a	2	5	-	-	-	-	1	11
50	a	-	-	-	-	-	-	-	12, 20, 23, 25, 26, 27, 28

PROBLEMA 25

FONTE ORIGINAL[1]: Davis e Patterson

RECURSO	R1	R2	R3
DISPONIBILIDADE	6	6	6

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
1	-	-	-	-	-	-
2	a	5	3	5	2	1
	b	6	3	3	1	
	c	7	2	4	1	
3	a	5	5	4	3	1
	b	6	4	4	3	
	c	8	3	3	3	
4	a	3	5	2	2	1
	b	4	4	2	2	
	c	6	3	2	2	
5	a	4	4	1	4	2, 3
	b	5	3	1	4	
	c	6	3	2	2	
6	a	1	5	5	4	4
	b	3	4	4	4	
	c	5	3	3	3	
7	a	6	3	5	2	4
	b	7	3	4	2	
	c	8	3	3	3	
8	a	2	2	4	4	2, 3
	b	3	2	3	4	
	c	4	2	3	3	
9	a	6	3	2	2	5, 7, 8
	b	7	2	2	2	
	c	8	2	1	1	
10	a	3	3	2	4	6, 7, 8
	b	4	2	2	4	
	c	6	2	2	2	
11	a	3	3	3	2	6
	b	4	2	3	2	
	c	5	2	2	2	
12	a	1	4	1	4	5

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
	b	2	3	1	4	
	c	3	3	1	3	
13	a	3	1	4	4	9, 10
	b	4	1	3	3	
	c	6	1	2	2	
14	a	6	2	2	2	12
	b	7	1	2	2	
15	a	3	5	5	4	9
	b	5	4	4	4	
	c	7	3	4	3	
16	a	4	1	5	4	11
	b	5	1	4	4	
	c	6	1	4	3	
17	a	4	4	5	4	15, 16
	b	6	3	4	4	
	c	8	3	3	3	
18	a	3	3	2	3	14
	b	4	2	2	3	
19	a	3	5	3	3	13
	b	4	4	3	3	
	c	6	3	3	3	
20	a	1	2	4	6	16
	b	2	2	4	5	
	c	4	2	4	4	
21	a	4	1	6	2	18, 19
	b	5	1	5	2	
	c	7	1	4	2	
22	a	6	3	2	1	17, 20
	b	7	2	2	1	
23	a	4	1	-	4	19
	b	5	1	-	3	
24	a	1	2	2	1	17
	b	2	1	2	1	
25	a	3	-	1	3	21, 22, 23, 24
	b	4	-	1	2	
26	a	3	2	2	2	23
	b	4	1	2	2	
27	a	-	-	-	-	25, 26

7

PROBLEMA 26

FONTE ORIGINAL: R.A. Jones (1984)

RECURSO	R1	R2	R3
DISPONIBILIDADE	*1*	*9*	*6*

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
1	a	5	-	5	6	-
	b	6	-	4	5	
	c	7	-	4	4	
2	a	10	1	3	-	1
	b	11	1	2	-	
	c	12	-	2	1	
3	a	14	-	3	2	1
	b	15	-	2	1	
4	a	12	1	4	3	1
	b	13	1	3	2	
	c	15	-	4	3	
5	a	15	-	5	2	2

ATIVIDADE	MODO[2]	DURAÇÃO	DEMANDA DE RECURSOS			ANTECESSORAS
			R1	R2	R3	
	b	16	-	3	2	
	c	17	1	3	1	
6	a	16	1	4	4	3
	b	17	1	3	3	
	c	18	-	5	6	
7	a	14	1	5	2	4
	b	15	1	3	2	
8	a	12	1	5	4	5
	b	13	1	4	3	
	c	15	-	5	4	
9	a	19	-	5	6	6
	b	21	-	4	5	
10	a	12	-	5	3	7
	b	13	-	4	3	
	c	14	-	4	2	
11	a	6	-	5	6	8, 9, 10
	b	7	-	4	5	
	c	9	-	3	4	