PLANEAMENTO DE PROJECTOS COM RECURSOS LIMITADOS

JORGE MAGALHÃES MENDES

POLITÉCNICO DO PORTO

Edições POLITEMA

POLITÉCNICO DO PORTO

FIPP FUNDAÇÃO POLITÉCNICO DO PORTO

Edições POLITEMA

EDIÇÕES POLITEMA

EDIÇÕES RECENTES

Condição feminina no império colonial português
Clara Sarmento (Coord.), 2008

Piscinas: tratamento de águas e utilização de energia
Vitorino de Matos Beleza, Rosária Santos, Marta Pinto, 2007

ARTES DO ESPECÁCULO

Quem matou Ambrósio?
José Topa e Claire Binyon, 2006

Miragem e Sedução: textos sobre cinema
José Coutinho e Castro, 2003

ARTES VISUAIS

Pintura Portuguesa Contemporânea: Colecção Instituto Politécnico do Porto
Fátima Lambert, 2005

Sobral Centeno
Sobral Centeno, Fátima Lambert, 2004

EDUCAÇÃO

Tecnologias, informação e educação
Paulo Ferreira, 2006

E-learning e formação avançada
Carlos Vaz de Carvalho (Org.), 2006

Implementação do processo de bolonha a nível nacional, por áreas de conhecimento – engenharia: relatório preliminar
Luís de Jesus Santos Soares, 2005

Curso de professores do 1.º ciclo do ensino básico – *curso de educadores de infância*
Luís de Jesus Santos Soares, 2003

Escola, pais e comunidade
Costa Matos, J. Vargas Pires, 1994

Problematizando a escola básica
António Abreu Pereira, Glória Souto, António Souto, José Maria Ribeiro, Abílio José Pires, 1994

Organização pedagógica da escola básica
Clara Oliveira, Fernanda Costa, Filomena Matos, João Carvalho, 1993

ENGENHARIA E TECNOLOGIA

Tratamento de águas de caldeiras
Francisco Teixeira, Isabel Pereira, Rosária Santos, Vitorino de Matos Beleza, 2001

Hipertexto/hipermedia
Armando Silva, 1996

Tratamento de águas de arrefecimento
Francisco Teixeira, Isabel Pereira, Rosária Santos, Vitorino de Matos Beleza, 1996

GESTÃO

Organização e Técnicas Empresariais
Isabel Ardions, Zita Romero e Arminda Sequeira, 2006

LITERATURA

Propostas de Leitura para "Generation X"
Teresa Tudela, 2004

MULTIMÉDIA

Glossário de termos multimédia: inglês/português
José Coutinho e Castro, 1997

MÚSICA

A Theory of Harmonic Modulation: The Plastic Model of Tonal Syntax and the Major-Minor Key System
J. Miguel Ribeiro-Pereira, 2005

Origens e Segredos da Música Portuguesa Contemporânea: música em som e imagem
Cândido Lima, 2003

Antologia Poético-Musical: textos traduzidos o mais literalmente possível de obras para Canto e Piano
Fernando Jorge Azevedo, 2002

PUBLICAÇÕES PERIÓDICAS

Polissema: revista de letras do ISCAP 2007 n.º 7
Cristina Pinto da Silva e Clara Sarmento, 2007

EDIÇÕES POLI_FONIA

DISCOS

AD LIBITUM
António Augusto Aguiar, 2005

10 Anos da Sinfonieta
Vários, 2006

PLANEAMENTO DE PROJECTOS COM RECURSOS LIMITADOS

Jorge Mendes

Edições Politema – Instituto Politécnico do Porto
Porto 2008

Biblioteca Nacional – Catalogação na Publicação

MENDES, Jorge Magalhães, 1963–

Planeamento de Projectos
com Recursos Limitados. – (Técnica)
ISBN 978-972-8688-49-0

CDU 658

Planeamento de projectos com recursos limitados
Jorge Magalhães Mendes

Fundação Instituto Politécnico do Porto
Porto, Dezembro de 2008

Colecção Técnica

Contactos:
Rua Dr. Bernardino de Almeida, 537
4200 – 072 Porto
Telefone 228302555
Fax 22 830 25 56
politema@ipp.pt

Edição Politema

Revisão
Pedro Sousa

Capa
Paulo Magalhães

Paginação
Débora Pinguinha

Fotografia do autor
A. Gorgal (Serviços Fotográficos do IPP)

Impressão
Sersilito Empresa Gráfica, Lda.

ISBN
978-972-8688-49-0

Depósito legal
285451/08

Tiragem
500 exemplares

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

PRÓLOGO

O interesse pela área da gestão de projectos, e em particular pelo planeamento, surgiu quando estava a fazer o meu estágio profissional numa empresa privada, no momento em que um responsável pelo planeamento da produção me desafiou a resolver o seu problema: elaborar planos realistas de afectação de tarefas a recursos (máquinas), dispensando um conjunto de documentos elaborados pela Informática central que, no seu entendimento, não lhe eram úteis.

O objectivo do estágio não era resolver este problema... mas permaneceu o interesse.

Quatro anos depois estava a entregar a minha dissertação de Mestrado que tinha uma proposta de solução para o problema.

Tratando-se de uma área do conhecimento de elevada complexidade e também de grande actualidade na indústria, permaneceu o interesse em continuar a investigação na área, realizando uma tese de doutoramento na área da gestão de projectos que contou com a orientação do Professor Doutor José Fernando Gonçalves e a co-orientação do Professor Doutor António Carvalho Brito, da Faculdade de Engenharia da Universidade do Porto.

A publicação desta obra surge neste contexto, tendo a expectativa que possa ser útil a todos os que procuram novas soluções para os seus problemas de planeamento de projectos com recursos limitados.

Jorge Mendes

INTRODUÇÃO

1.1 Relevância do problema

A importância da gestão de projectos tem vindo a aumentar, em particular desde o seu reconhecimento como actividade fundamental nas organizações sujeitas a concorrência numa economia de mercado. Esta importância reflecte-se num reforço da capacidade de concorrência da organização, num aumento de produtividade e num aumento de qualidade.

A gestão de projectos compreende um complexo processo de tomada de decisões, particularmente ao nível do planeamento e programação, na qual é dada ênfase especial ao cumprimento das datas de entrega e dos valores orçamentados.

O planeamento geral envolve a definição das actividades, do conjunto de recursos, das restrições de dependência entre actividades e das quantidades de recursos necessárias à realização das actividades que constituem os projectos.

O planeamento operacional ou programação envolve a determinação dos tempos de início e de fim de cada actividade de cada projecto, considerando as definições realizadas na fase de planeamento geral. Ao resultado obtido pelo planeamento operacional designa-se de plano base.

No planeamento de projectos são frequentemente utilizados os métodos clássicos seguintes: gráfico de Gantt, método CPM (*Critical Path Method*) e método PERT (*Program Evaluation and Review Technique*). Apesar da sua grande divulgação estes métodos têm sérias limitações pois assumem que os recursos são ilimitados e aplicam-se apenas a um só projecto e não a múltiplos projectos.

1.2 Conceito de projecto

A palavra "projecto" tem diferentes significados para diferentes pessoas. Nenhuma definição de projecto se adequa a toda e qualquer situação, mas a definição que surgiu com a ISO 8402 tem vindo a ganhar uma grande aceitação por um crescente número de utilizadores:

Projecto processo único, consistindo num conjunto de actividades coordenadas e controladas, com datas de início e de fim, que concorrem para realizar objectivos, de acordo com necessidades específicas, nomeadamente restrições de tempo, de custo e de recursos.

A norma ISO 10006 – *Guideline to Quality in Project Management* adiciona algumas notas para realçar aspectos do uso do termo "projecto":

- A organização é temporária e criada para o ciclo de vida do projecto;
- Algumas vezes um projecto é uma parte da estrutura de um grande projecto;
- Os objectivos do projecto e as características do produto podem ser definidas e alcançadas progressivamente durante o desenvolvimento do projecto;
- O resultado de um projecto pode ser a criação de uma ou várias unidades de um produto;
- As inter-relações entre actividades de um projecto podem ser complexas.

Os projectos têm características muito diferentes na sua realização. O primeiro projecto a ser realizado duma forma que poderemos classificar como tendo preocupação de gestão, foi o projecto Manhattan responsável pelo desenvolvimento da primeira bomba atómica. Contudo, a história está repleta de projectos, alguns deles bem conhecidos de todos nós: as Pirâmides do Egipto, a construção do palácio do Rei Minos em Creta, a construção do Partenom na Grécia, a construção dos templos Maia na América Central, a construção da Grande Muralha da China, entre outros. Nos anos cinquenta e sessenta os métodos formais de gestão de projectos tiveram um grande impulso, particularmente com o programa dos Mísseis Polaris, projecto Apolo e vários programas de defesa nos Estados Unidos da América. Nos dias de hoje os projectos têm muitas e variadas formas de que são exemplo:

- Construção do túnel da Mancha;
- Renovação do Aeroporto Francisco Sá Carneiro;
- Construção de uma ponte, auto-estrada ou edifício;
- Abertura de um armazém;
- Projecto de um novo avião (Airbus A380);
- Reformulação da implantação de uma fábrica;
- Implementação de um novo sistema de informação;
- Escrever uma tese;
- ...

Todos os projectos, em maior ou menor extensão, têm algumas características em comum, Slack *et al.* (1998), nomeadamente:

Um objectivo quantificável num produto final, resultado ou serviço que é tipicamente definido em termos de custo, qualidade e tempo, resultado da realização das várias actividades de um projecto;

Unicidade um projecto é único, isto é, não é repetitivo. Mesmo que o projecto possa parecer o mesmo, como por exemplo a construção de uma fábrica de produtos químicos, tendo inclusive as mesmas especificações, tem necessariamente diferenças em termos de utilização de recursos e condições ambientais no qual o projecto se realiza;

Complexidade as relações entre as várias actividades que têm de ser realizadas para se obterem os objectivos podem ser muito complexas, por exemplo, especificar intervalos de tempo mínimos e máximos entre tempos de fim e de início entre actividades;

Natureza temporária os projectos têm uma data definida de início e de fim, o que significa que uma grande quantidade de recursos tem de ser concentrada para levar a termo a realização do projecto. Frequentemente organizações temporárias são criadas, por exemplo consórcio para a construção de grandes projectos de construção civil, para em conjunto terem os meios necessários para a realização de um projecto;

Incerteza os projectos são planeados antes de serem executados e por esse facto têm associado um elemento de risco: a incerteza do tempo de realização;

Ciclo de vida ao longo do seu ciclo de vida um projecto passa por diferentes fases. Para começar existe a fase conceptual de projecto na qual a organização determina que um projecto necessita de ser realizado ou recebe um pedido de um cliente que propõe um plano para a sua realização. De seguida existe a fase de definição de projecto na qual os objectivos e os trabalhos a realizar são definidos e é decidido como a organização vai conseguir realizar e satisfazer os objectivos e as medidas de desempenho. Após um projecto ser devidamente definido e aprovado, começa a fase de planeamento. A fase de planeamento geral envolve a divisão do projecto em quantidades de trabalho que consistem em actividades específicas as quais têm de ser realizadas por forma a cumprir os diferentes objectivos do projecto. Nesta fase devem ser estimadas as durações das actividades, as necessidades e disponibilidades de recursos e definidas em detalhe as inter-relações entre as actividades. O projecto entra de seguida na fase de planeamento operacional ou programação na qual se constrói um plano base admissível em termos das restrições de precedência entre actividades e da capacidade disponível dos recursos, identificando os tempos programados de início e de fim de cada actividade. De seguida o projecto deve ser tornado uma realidade. Durante a execução do projecto, o seu progresso deve ser monitorizado e acções correctivas devem ser tomadas sempre que necessário. Finalmente, o projecto entra na fase de conclusão que envolve a entrega do resultado (produtos e/ou serviços).

Na literatura é possível encontrar algumas definições de projecto que realçam a importância da unicidade, como sendo a característica mais importante. Independentemente da maior ou menor importância das características de um projecto, numerosos e diversos problemas que ocorrem no contexto do planeamento de actividades ou operações (planeamento com capacidade finita), partilham um número de características que os tornam adequados à utilização das técnicas desenvolvidas no contexto do planeamento de projectos. Como consequência, vários tipos de problemas estudados na teoria do planeamento (problemas de planeamento de operações numa máquina, planeamento de operações em máquinas paralelas, *flow shops*, *job shops*, etc.) podem ser modelados como subproblemas ou variantes do problema básico de planeamento de projectos com recursos limitados RCPSP (*Resource Constrained Project Scheduling Problem*), tema que vai ser estudado ao longo deste trabalho.

1.3 Gestão de projectos

A gestão de projectos tem tido um papel cada vez mais um importante na sociedade, particularmente nas últimas décadas. No mundo actual, em que a competitividade é uma característica fundamental, é determinante cumprir os prazos de entrega previstos, com o orçamento estimado e com a qualidade pretendida. Não é por isso surpreendente que a gestão de projectos seja um importante tema, ao qual se dedica cada vez mais importância.

Sendo assim, descrevem-se de seguida as principais fases da gestão de projectos.

1.3.1 A fase conceptual

Na fase conceptual identifica-se qual a necessidade a ser satisfeita, podendo tratar-se de um novo produto ou serviço, uma mudança de instalações de um local para outro, um novo sistema de informação, uma campanha de marketing, etc. A identificação da necessidade, problema ou oportunidade, pode resultar das solicitações dos clientes, de uma equipa de projecto ou de empreendedores.

Na fase conceptual existem apenas as linhas gerais do problema a ser resolvido. Com frequência são realizados estudos de viabilidade com vista a clarificar o projecto antes de avançar para a fase seguinte.

1.3.2 A fase de definição

Antes de ter início a complexa tarefa de planear e executar o projecto é necessário definir claramente o que vai ser desenvolvido na solução proposta. Basicamente

são três os elementos necessários para definir um projecto: objectivos, âmbito e estratégia:

Objectivos Devem ser claros e mensuráveis. Os três objectivos fundamentais da gestão de projectos envolvem normalmente a minimização ou maximização de medidas associadas aos critérios: tempo, custo e qualidade. A importância relativa de cada objectivo difere com o tipo de projecto. Por exemplo, no desenvolvimento de um novo produto ou de um produto inovador o tempo é normalmente considerado como medida de desempenho mais relevante;

Âmbito Identifica os trabalhos a realizar e os resultados a obter (produto ou serviço). É no âmbito do projecto que se deve definir claramente o que vai e o que não vai ser feito durante o projecto. O âmbito estabelece limites de forma a que o cliente saiba o que é esperado quando o projecto estiver concluído;

Estratégia Define de forma geral como a organização pode concretizar os objectivos do projecto e satisfazer as medidas de desempenho. Os factores económicos, tecnológicos, legais, geográficos e sociais que afectam a execução de um projecto devem ser claramente identificados e examinados. Após todos os factores estarem identificados em detalhe, a equipa de projecto pode desenvolver uma série de estratégias por forma a definir o que vai ser realizado e como vai ser realizado.

Em geral o objectivo principal do problema do planeamento e programação de projectos consiste na minimização da duração do projecto. Contudo, a maioria dos problemas reais consiste em vários projectos em simultâneo e está sujeita a múltiplos objectivos, que entram com frequência em conflito. Outros objectivos incluem a minimização do custo, a maximização do valor presente do projecto, a utilização dos recursos, a eficiência dos recursos, o número de datas de entrega cumpridas ou não e a minimização dos em curso de fabrico.

Os conflitos entre objectivos acontecem com frequência. Por exemplo, pode ser fácil reduzir a duração de um projecto afectando recursos mais caros às actividades, contudo estaríamos a aumentar o custo do projecto.

1.3.3 A fase de planeamento geral

A fase de planeamento geral, designada a partir de agora simplesmente por planeamento, tem início após a definição do projecto. O processo de planeamento envolve as seguintes etapas fundamentais:

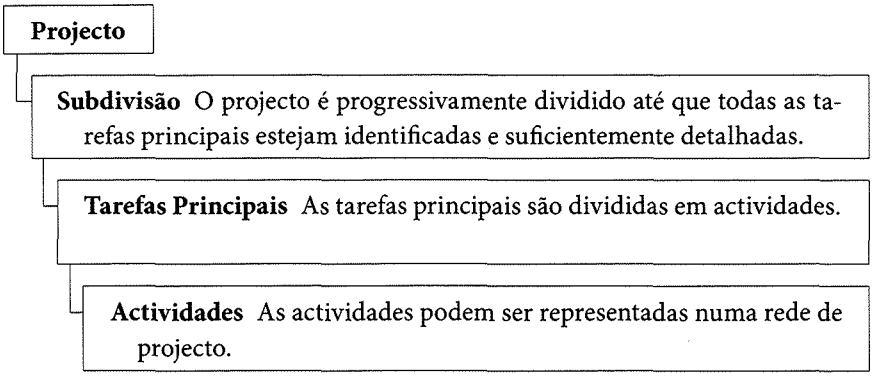
- Identificar as actividades;
- Estimar as durações das actividades;

- Definir as dependências entre actividades;
- Identificar os recursos;
- Definir as capacidades disponíveis dos recursos.

1.3.3.1 Actividades

A maioria dos projectos são demasiadamente complexos para serem planeados e controlados eficazmente, sem que sejam previamente subdivididos em partes gerivéis usando a estrutura de divisão de trabalho, (WBS - *Work Breakdown Structure*). O objectivo do WBS é o de dividir o projecto em tarefas principais e identificar as actividades específicas que necessitam de ser realizadas para cada tarefa principal por forma a cumprir os objectivos do projecto, ver **fig. 1.1**.

fig. 1.1 Processo da estrutura de divisão de trabalho (WBS).



A designação "actividade" pode compreender um trabalho qualquer a executar: furar, pintar, um tempo de espera, etc., e é definida por dois acontecimentos que marcam o início e o fim da actividade. Os acontecimentos não consomem tempo nem recursos de qualquer espécie.

A uma actividade associa-se sempre uma duração, cujo valor depende dos recursos ou meios disponíveis para a concretizar.

Os problemas podem permitir um ou mais tipos de dependências entre actividades:

- Início-Início;
- Início-Final;
- Final-Início;
- Final-Final.

A execução de uma actividade pode ou não ser interrompida, sendo mais frequente o último caso.

1.3.3.2 Recursos

Um recurso é um meio (mão-de-obra, máquinas, dinheiro, etc.) que está disponível para a execução de uma ou mais actividades.

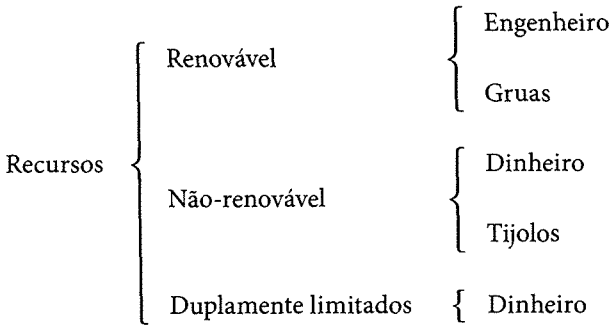
Cada um dos recursos a serem identificados pode ser classificado numa das seguintes três categorias (ver **fig. 1.2**):

Renovável Quando está limitado apenas de período a período. Independentemente da duração de cada projecto cada recurso está disponível para cada período (hora, dia, turno, semana, mês, etc.). A mão-de--obra e equipamento são exemplos de recursos renováveis;

Não-renovável São recursos que estão apenas limitados ao longo de todo o projecto. Neste caso não são impostos limites por período. Exemplos deste tipo de recurso são o orçamento disponível, matérias primas (ex.: 100 kg de cimento), etc.;

Duplamente limitado São recursos para os quais existem limites período a período e ao longo de todo o projecto, por exemplo, poder-se-á ter disponível um orçamento ao longo de todo o projecto de um milhão de euros, mas semanalmente não poder exceder cento e cinquenta mil euros.

fig. 1.2 Classificação dos recursos.



A diferença entre recursos renováveis e não-renováveis pode ser ténue. Por vezes, os recursos renováveis podem tornar-se não-renováveis e vice-versa.

Os recursos variam em capacidade e custo. Alguns recursos incluem restrições temporais que limitam o seu período de utilização, por exemplo, uma equipa de

maquinistas pode estar disponível apenas um turno por dia. As restrições podem ser ainda mais complicadas: uma equipa de maquinistas pode estar disponível para eventuais horas extras com a condição de terem um dia de folga por cada dois dias de trabalho extra.

1.3.3.3 Restrições

Existem essencialmente dois tipos de restrições: as de precedência entre actividades e as relativas à utilização de recursos. As restrições de precedência definem dependências entre actividades. Por exemplo a pintura de uma parede só pode ter início após o reboco estar concluído e seco. As restrições relativas aos recursos limitam a utilização destes ao longo de um ou vários períodos de tempo.

As actividades de um projecto podem ser representadas graficamente sob forma de uma rede. A rede de projecto ilustra as dependências entre as actividades.

Finalmente, após a identificação das actividades, dependências entre actividades, tipos de recursos e capacidades de recursos, devem ser estimados os custos.

1.3.4 A fase de planeamento operacional

A fase de planeamento operacional ou programação envolve a construção de um plano base o qual especifica as datas de início e de fim para todas as actividades.

Um plano diz-se admissível se satisfaz todas as restrições. Um plano diz-se óptimo se satisfaz todas as restrições e é também tão bom ou melhor que qualquer outro plano admissível. A qualidade de um plano é definida pelas medidas associadas aos objectivos.

O desenvolvimento de um plano base bem concebido é crítico para o sucesso do projecto.

1.3.5 A fase de controlo

Uma vez aceite um plano base este deve ser implementado. A sua implementação envolve a execução do trabalho, de acordo com o plano base e o controlo, do trabalho realizado, com vista a que o projecto seja concluído cumprindo os objectivos. Uma vez iniciado o projecto, o progresso deve ser monitorado o que envolve a difícil tarefa de medir o trabalho realizado e compará-lo com o trabalho previsto. Se esta comparação revelar que o projecto se começa a atrasar, ou

a exceder o orçamento, ou a não cumprir algumas das especificações originais, devem ser de imediato tomadas acções correctivas.

1.3.6 A fase de conclusão

A fase de conclusão é a última fase do projecto. Esta fase é tão importante como a inicial. Um projecto não deve deixar de cumprir os objectivos. A gestão de projectos, em particular o seu planeamento, tem como primeiro e último objectivo a satisfação do cliente.

1.4 Definição do problema

O problema objecto deste trabalho consiste na geração de planos base de boa qualidade pois conforme mencionado anteriormente a utilização de um bom plano base é fundamental para a boa execução do projecto e o cumprimento dos objectivos.

A construção de um plano base é realizada na fase de planeamento operacional. O processo de obtenção de planos base, representado na **fig. 1.3**, consiste nas seguintes três etapas:

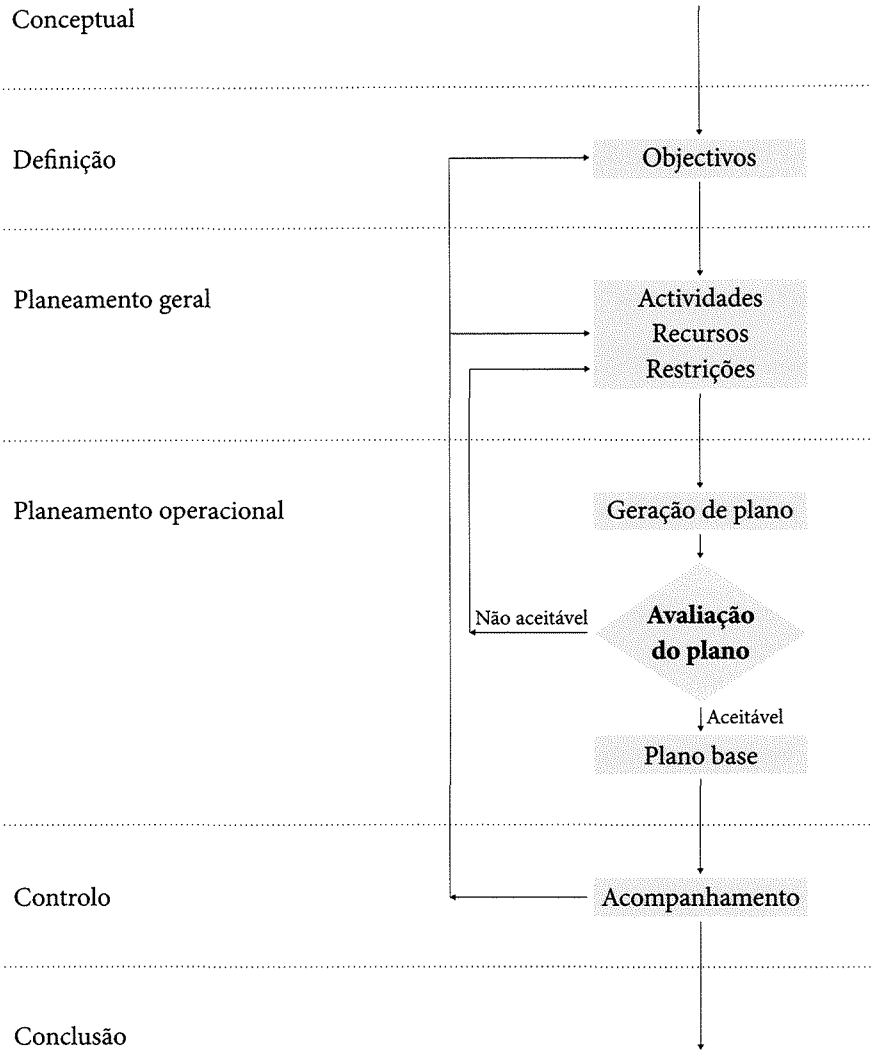
Definição/alteração de objectivos, actividades, recursos, restrições e planos;
Geração de planos;
Avaliação de um plano.

A definição de objectivos, actividades, recursos e restrições é inicialmente realizada na fase de planeamento geral, contudo na fase de planeamento operacional ou programação poderá ser necessário rever a definição inicial, com vista a poder obter planos base aceitáveis.

A geração de um ou vários planos deve considerar as restrições definidas na fase do planeamento. A validação de um plano consiste em determinar se o plano proposto cumpre os objectivos. A não validação do plano significa que algum objectivo não está a ser cumprido, pelo que será necessário rever alguns dos dados das fases anteriores ao planeamento operacional.

A revisão de um plano pode implicar a alteração dos valores definidos no planeamento, de forma a ser possível obter um plano mais adequado. Os valores que podem ser alterados são, por exemplo, o modo de execução de uma actividade ou algumas restrições, por exemplo, relações de dependência ou capacidade de recursos.

fig. 1.3 Fases de um projecto.



A geração de um plano base será tanto mais complexa quanto maior for o número de actividades, o número de recursos e a quantidade de restrições.

A validação do plano gerado será tanto mais difícil quanto maiores forem os elementos a analisar com vista a obter um plano óptimo ou satisfatório.

Obviamente que objectivos mal definidos ou um planeamento geral mal conseguido contribuirão igualmente para aumentar a dificuldade da fase de planeamento operacional.

A geração de um bom plano base é essencial para se prosseguir com sucesso nas fases da gestão de projectos.

Para além dos elementos já descritos que dificultam a obtenção de um plano base, de que são exemplo os objectivos identificados na fase de definição ou as restrições na fase de planeamento, outros elementos podem aumentar de forma significativa a complexidade do problema:

Número de projectos simultâneos Variável de organização para organização. Quando o problema de planeamento operacional de projectos envolve um único projecto designa-se por problema RCPSP (*Resource Contrained Project Scheduling Problem*) e quando envolve vários projectos simultaneamente designa-se por problema RCMPSP (*Resource Contrained Multi-Project Scheduling Problem*);

Natureza das informações do projecto Quando os dados (por exemplo, a sua duração) sobre as actividades e os recursos envolvidos são determinados com precisão, diz-se que o problema é de natureza determinística; pelo contrário, quando alguns dos principais dados são determinados com incerteza diz-se que o problema é de natureza probabilística. A utilização de variáveis com incerteza aumenta a complexidade do problema;

Tipos de objectivo A utilização de diferentes tipos de objectivo, por exemplo um do tipo temporal e outro financeiro, pode ser um factor de grande dificuldade na validação de um plano base, na medida em que reduzir a duração de um projecto pode implicar aumento de custos, i.e., são objectivos conflitantes entre si, pelo que se torna indispensável uma decisão de compromisso entre eles para uma solução final aceitável.

1.5 Conclusões

As novas tendências de mercado, particularmente com o mercado único europeu e a globalização, exigem das organizações uma utilização eficiente e eficaz dos recursos (humanos, financeiros, equipamentos, etc..) por forma a garantir o aumento da sua competitividade e muitas vezes a sua própria sobrevivência.

Alguns dos aspectos que tornam relevantes o investimento no planeamento operacional apresentam-se de seguida:

1 Os produtos e/ou serviços tendem a ser cada vez mais personalizados. Este facto traduz-se numa rápida transformação dos sistemas de produção do tipo repetitivo para sistemas de produção do tipo projecto;
2 O cumprimento dos prazos de entrega é cada vez mais um factor competitivo para as empresas, pelo que se torna necessário investir em sistemas de planeamento, que tenham em conta este factor de competitividade;
3 Executar os projectos nas datas evitando atrasos que podem significar fortes penalizações ou inclusivamente perda de clientes;
4 Não iniciar os projectos mais cedo do que o necessário pois tal implica a mobilização de recursos;
5 Não concluir os projectos antes das datas previstas por forma a evitar os custos resultantes do armazenamento e mobilização de capital.

2.1 Introdução

A área da teoria do planeamento operacional é muito dinâmica e reconhecidamente começou, no início dos anos 50, com o trabalho pioneiro de Johnson (1954) sobre programação *flow shop* de duas máquinas. Este trabalho, juntamente com os trabalhos de Jackson e Smith, formam a base da teoria da programação de máquinas, MacCarthy e Liu (1993).

Desde meados da década de 50 as técnicas padrão da gestão de projectos PERT (*Program Evaluation and Review Technique*) e CPM (*Critical Path Method*), vêm sendo largamente empregadas. Estes métodos assumem que os recursos necessários à execução dos projectos estão disponíveis em quantidades ilimitadas. Contudo, na prática, estes recursos estão disponíveis em quantidades limitadas. Como consequência tem aumentado o número de trabalhos de investigação sobre o problema de planeamento de projectos com recursos limitados.

Nas próximas secções faz-se uma revisão da literatura para os problemas do planeamento operacional de um único projecto, de múltiplos projectos e de sistemas do tipo *job shop*.

2.2 A Revisão da literatura – RCPSP

Blazewicz *et al.* (1983) demonstraram que o problema RCPSP é uma generalização do problema clássico do JSP (*Job Shop Problem*) o qual pertence à classe de problemas do tipo NP-difícil, justificando assim o indispensável recurso a algoritmos baseados em heurísticas, para a resolução de problemas reais em tempo útil. Tal facto tem levado a que, particularmente nas últimas duas décadas, uma parte do esforço de investigação se exerça na área dos algoritmos heurísticos.

Nas secções seguintes apresentam-se os seguintes métodos para a resolução do problema RCPSP: representação gráfica, técnicas de programação com redes, os métodos exactos e os métodos aproximados.

2.2.1 Representação gráfica

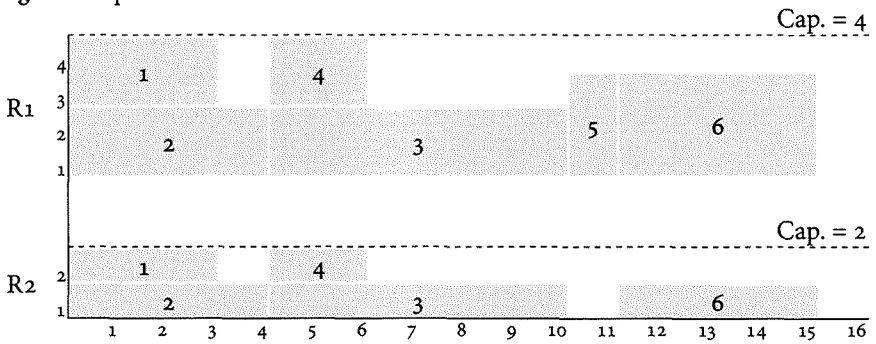
O mapa de Gantt e a rede do projecto são representações que descrevem graficamente o problema do planeamento operacional de projectos.

2.2.1.1 Mapa de Gantt

De acordo com Harding (1989), o mapa de Gantt foi idealizado pelo engenheiro industrial norte-americano Henry Gantt em 1917, tendo sido aplicado na área militar durante a primeira guerra mundial. Segundo Belchior (1970), a grande contribuição deste mapa para a arte da decisão foi a de relacionar os factos com o tempo.

O mapa de Gantt consiste num gráfico no qual o eixo dos *x* representa o tempo e o eixo dos *y* representa os recursos. Cada actividade é representada por um rectângulo, na qual a largura corresponde à duração da actividade e a altura corresponde à quantidade utilizada, por unidade de tempo, do recurso onde se encontra colocada. Na **fig.2.1** apresenta-se um exemplo de um mapa de Gantt na qual existem 6 actividades (1,2,3,4,5,6) e dois recursos (R1,R2) cujas capacidades são respectivamente 4 e 2. É fácil verificar que a actividade 3 tem início no instante 4 e termina no instante 10 e que gasta ao longo desse período 2 unidades de R1 e 1 de R2. De igual modo, pode-se verificar que a actividade 5 tem início no instante 10 e termina no instante 11 e que ao longo da sua duração gasta 3 unidades de R1.

fig. 2.1 Mapa de Gantt.



Embora o mapa de Gantt ilustre razoavelmente bem o início e fim das actividades, a utilização de recursos e as capacidades livres, não apresenta as interdependências entre as actividades.

2.2.1.2 Rede de projecto

Um projecto pode ser representado por uma rede. Uma rede é constituída por um conjunto de nós e por um conjunto de arcos. Existem dois tipos de representação de uma rede de projecto:

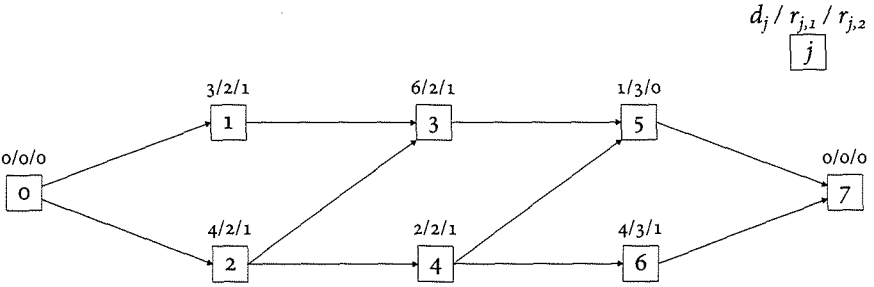
AOA (*Activity-on-arrow*) Actividades representadas nos arcos enquanto os acontecimentos estão representados nos nós;

AON (*Activity-on-node*) Actividades representadas nos nós enquanto cada arco representa uma relação de precedência entre duas actividades.

No tipo de representação AON utiliza-se uma actividade inicial fictícia com duração nula como início do projecto e utiliza-se também uma actividade final fictícia com a duração nula como fim do projecto.

Na **fig. 2.2** ilustra-se a rede de projecto do tipo AON relativa ao exemplo da **fig. 2.1**. As actividades são representadas por j, as durações destas por d_j e a utilização do recurso i pela actividade j por $r_{j,i}$. Neste exemplo as actividades inicial e final de projecto são respectivamente a actividade 0 e a actividade 7.

fig. 2.2 Exemplo de rede de projecto com representação AON.



As redes de projecto têm como vantagem sobre o mapa de Gantt ilustrar as dependências entre actividades. O mapa de Gantt tem como vantagem uma fácil leitura do início e fim das actividades e da utilização dos recursos. O mapa de Gantt e a rede do projecto devem ser utilizados em conjunto.

2.2 Técnicas de programação com redes

A base das técnicas de programação com redes são os diagramas de rede do projecto Davis (1973), com os seus dois tipos de representação. Os diagramas de redes do projecto são a base dos métodos tradicionais CPM e PERT.

2.2.1 CPM e PERT

O CPM (*Critical Path Method*) foi criado em 1957, pela C.ia E.U. du Pont, assessorada pela Remington Rand Division da Sperry Rand Corporation, com o objectivo

de melhorar a programação de projectos ligados às fábricas de produtos químicos. Saliente-se que as actividades pertencentes a estes projectos tinham durações estimadas com precisão considerável, facto que resultou na formulação determinística deste método.

No ano seguinte, em 1958, um grupo formado pelo Special Projects Office da marinha norte-americana, pela firma consultora de empresas Booz-Allen & Hamilton International, e pela empresa de projécteis balísticos Lockheed, conceberam o PERT (*Program Evaluation and Review Technique*) com o objectivo de planear a fabricação do míssil Polaris. Neste projecto havia milhares de actividades diferentes para as quais não era conhecida a sua duração pelo facto de nunca terem sido antes produzidas, o que motivou a introdução do conceito de duração probabilística das actividades.

A forma de estimar as durações das actividades no CPM e no PERT representa a diferença fundamental entre as duas técnicas: enquanto no CPM temos uma duração determinística, no PERT temos uma duração probabilística.

De seguida apresenta-se o processo de cálculo das durações das actividades na técnica do PERT.

A técnica PERT incorpora incerteza na duração das actividades, utilizando 3 parâmetros para estimar a duração de cada actividade. Os parâmetros são:

- ***a*** **Duração optimista** é a menor duração que pode ser esperada se tudo correr excepcionalmente bem;
- ***b*** **Duração pessimista** é a pior duração que pode ser esperada se tudo correr mal;
- ***m*** **Duração mais provável** é a melhor duração ou valor da expectativa.

A duração probabilística das actividades ou tempo esperado, *Te*, e a respectiva variância, σ^2, são calculadas da seguinte forma:

$$Te = (a + 4m + b)/6$$

$$\sigma^2 = \left(\frac{b-a}{6}\right)^2.$$

As fórmulas para o cálculo de *Te* e σ^2, são baseadas na distribuição de probabilidades beta.

2.3 Métodos exactos

2.3.1 Modelos de programação matemática

Os problemas de planeamento operacional podem ser formulados com programação matemática, particularmente programação linear e/ou inteira.

Pritsker *et al.* (1969) apresentam uma formulação baseada em programação inteira, na qual estabelecem uma ligação entre as variáveis de decisão binárias e os tempos de conclusão das actividades, que tem como objectivo minimizar o tempo de conclusão do projecto.

Finalmente, é de referir que a formulação descrita por Pritsker *et al.* necessita de n actividades, T instantes de tempo e m restrições de recursos para cada instante de tempo. Desta forma a formulação necessita de nT variáveis binárias e $n+n(n-1)/2+mT$ restrições, i.e., este modelo requer $O(n^2+mT)$ restrições.

Kaplan (1988) apresenta uma formulação para uma versão com interrupção das actividades para o problema RCPSP e requer $O(n^2T)$ restrições.

Klein (2000) apresenta uma formulação utilizando variáveis binárias para uma versão sem interrupção das actividades para o problema e requer $O(n^2T)$ restrições.

Uma outra formulação é apresentada por Alvarez-Valdés e Tamarit (1993) para uma versão sem interrupção das actividades. Esta formulação é baseada em conjuntos incompatíveis. Um conjunto incompatível é um conjunto de actividades entre as quais não existem relações de precedência, mas que podem violar as restrições de recursos se sequenciadas em paralelo. Este modelo requer $O(2^n)$ restrições.

Mingozzi *et al.* (1998) apresentam uma formulação alternativa para uma versão sem interrupção das actividades. Esta formulação é baseada no conceito de subconjuntos admissíveis, i.e., subconjuntos de actividades entre as quais não existem relações de precedência, e se as actividades são sequenciadas em paralelo, não violam as restrições de capacidade dos recursos. Este modelo requer $O(n^2, nT)$ restrições.

Várias outras formulações em programação linear, inteira ou dinâmica, para o problema da programação de projectos, têm sido propostas:

Linear e/ou inteira Patterson e Huber (1974), Doersch e Patterson (1977), Patterson *et al.* (1989), Talbot e Patterson (1978), Yang *et al.* (1992), Oguz e Bala (1994), Icmeli e Rom (1997), Brucker e Knust (2000) e Mohring *et al.* (2002);

Dinâmica Held e Karp (1962) e Tavares (1986).

Excelentes trabalhos de análise, sobre métodos exactos, foram realizados por Patterson (1984), Demeulemeester (1992) e Kolisch e Padman (2001).

O elevado número de variáveis, o número de restrições nos diversos tipos de formulação linear, inteira ou dinâmica, a dificuldade em adaptar os modelos reais à formulação matemática, tem impedido o desenvolvimento de procedimentos para problemas de grande dimensão, Davis (1985).

2.2.3.2 Modelos baseados em *branch and bound*

A técnica do *branch and bound* é uma das técnicas mais usadas nos métodos exactos para a solução do problema do planeamento operacional de projectos com recursos limitados.

Na área dos algoritmos óptimos tem sido proposto por vários autores um vasto conjunto de abordagens utilizando a técnica do *branch and bound*, salientam-se as contribuições de Johnson (1967), Shrage (1970), Hastings (1972), Stinson *et al.* (1978), Christofides *et al.* (1987), Bell e Park (1990), Carlier e Latapie (1991), Demeulemeester e Herroelen (1992), Demeulemeester e Herroelen (1997), Sprecher (1997), Brucker *et al.* (1998), Klein e Scholl (1998 a,b), Mingozzi *et al.* (1998), Reyck e Herroelen (1998), Schwindt (1998), Dorndorf *et al.* (2000a, b) e Zamani (2001).

A natureza exponencial do algoritmo *branch and bound*, tem impedido o desenvolvimento de procedimentos para problemas de grande dimensão.

2.2.4 Métodos aproximados

O problema RCPSP é NP-difícil. Por este facto os métodos exactos não são capazes de produzir soluções óptimas em tempo útil. A necessidade de obter planos de boa qualidade em tempo real levou à utilização de métodos aproximados. Excelentes trabalhos sobre métodos aproximados para o problema RCPSP podem ser encontrados em Kolisch e Hartmann (1999) e Hartmann e Kolisch (2000).

Os métodos aproximados podem ser divididos em duas categorias:

1 Métodos construtivos;
2 Métodos de busca local.

Os métodos construtivos começam com um plano vazio e adicionam sucessivamente actividades, uma a uma, até ser obtido um plano final. A ordem em que as actividades são adicionadas ao plano é definida por regras de prioridade.

Os métodos de busca local começam normalmente com um plano obtido por um método construtivo. Os métodos de busca local aplicam alterações sucessivas ao plano por forma a transformar este num plano de melhor qualidade. Estas operações são aplicadas até que seja alcançado um óptimo local. Para evitar o problema de parar num óptimo local foram desenvolvidas metaheurísticas tais como o *tabu search*, *simulated annealing* e os algoritmos genéticos.

2.4.1 Tipos de planos

Existem os seguintes tipos de planos:

Admissível é um plano que satisfaz as restrições de precedência entre actividades e as restrições de capacidade dos recursos;

Semi-activo é um plano admissível que assegura que cada actividade é iniciada o mais cedo possível (isto é, não existem tempos de inactividade desnecessários);

Activo é um plano admissível que assegura que nenhuma actividade pode ser iniciada mais cedo sem provocar um atraso noutras actividades ou violar as restrições tecnológicas ou de recursos. Os planos activos formam um subconjunto dos semi-activos, isto é, um plano activo é necessariamente semi-activo. De notar que a solução óptima pertence ao subconjunto dos activos, Sprecher *et al.* (1995);

Não-Atrasado é um plano admissível que assegura que nenhum recurso fica inactivo quando pode iniciar o processamento de alguma actividade. Estes planos fazem parte do conjunto dos activos, logo, também são semi-activos.

Estes tipos de planos aplicam-se aos problemas RCPSP, RCMPSP e JSP. Para além da classificação dos planos é importante estabelecer as relações de dominância entre os diferentes subconjuntos de planos.

As relações de domínio entre os diferentes subconjuntos são estabelecidas da seguinte forma (Baker, 1974 e Kolisch, 1995): o subconjunto dos semi-activos domina o conjunto dos planos admissíveis, o subconjunto dos activos domina o subconjunto dos semi-activos e o subconjunto dos não-atrasados não dominam o subconjunto dos activos, o que significa que para optimizar uma medida de desempenho regular é suficiente considerar apenas o subconjunto dos activos, ver **fig. 2.3**.

fig. 2.3 Espaço de soluções.

Geralmente o plano óptimo pertence ao subconjunto dos activos, embora, nalguns casos, pode pertencer ao subconjunto dos não-atrasados.

2.2.4.2 Métodos construtivos

Os métodos construtivos consistem em duas componentes fundamentais:

1 Esquema de geração de planos;
2 Regra de prioridade.

Um esquema de geração de planos determina a forma como um plano admissível é construído através da atribuição sucessiva de tempos de início e de fim às diferentes actividades de um projecto.

Os esquemas de geração de planos SGS (*Schedule Generation Schemes*) são a parte nuclear dos métodos construtivos para a geração de soluções do problema do RCPSP. Os esquemas de geração de planos SGS consideram dois esquemas base:

SGS Paralelo Baseado no incremento do tempo;
SGS Série Baseado no incremento do número de actividades programadas.

Os esquemas SGS Série e SGS Paralelo podem ser utilizados com diferentes direcções de programação: para a frente, para trás ou uma combinação das duas direcções que se designa por programação bi-direccional.

As regras de prioridade determinam a próxima actividade a ser seleccionada para programada. A aplicação das regras de prioridade permite obter uma ordenação através de uma lista de prioridade das actividades. A ordenação das activi-

dades deve ser realizada considerando as restrições entre actividades.

A aplicação de um esquema de geração de planos com uma direcção de programação combinado com uma regra de prioridade resulta nos métodos de passagem única. A combinação de vários esquemas de geração de planos com diferentes direcções de programação e várias regras de prioridade resulta nos métodos de multi-passagem.

Nas secções seguintes descrevem-se os esquemas de gerações de planos mais relevantes.

.2.4.2.1 Esquemas de geração de planos

Nas secções seguintes dá-se particular ênfase aos esquemas de geração de planos SGS Série e SGS Paralelo, referindo-se ainda a programação para trás e a programação bi-direccional.

.2.4.2.1.1 SGS Série

O método SGS Série foi proposto por Kelley (1963). O esquema de geração de planos baseado no SGS Série, que constrói planos activos, utiliza o incremento da actividade. O SGS Série consiste em *1,...,n* estágios, em cada um dos quais é programada uma actividade tendo em conta as relações de precedência das actividades e a capacidade dos recursos.

Cada estágio tem associado dois conjuntos. O conjunto das actividades programadas S_g, e o conjunto das actividades disponíveis para programação $D_g = \left\{ j \in J \setminus S_g \mid P_j \subseteq S_g \right\}$. Note-se que a reunião dos conjuntos S_g e D_g não representa todas as actividades J do problema, dado que existem actividades que ainda não estão disponíveis, designadas por actividades não-disponíveis, que não podem ser programadas no estágio g porque nem todas as suas actividades predecessoras foram programadas. Seja

$$RD_k(t) = R_k(t) - \sum_{j \in A((t))} r_{j,k}$$

a capacidade disponível do recurso k no instante t e seja $CF_g = \left\{ F_j \mid j \in S_g \right\}$ o conjunto de todos os tempos de fim das actividades já sequenciadas.

A inicialização do algoritmo começa por atribuir à actividade $j=0$ um tempo de conclusão igual a zero. No início de cada estágio, o conjunto das actividades disponíveis D_g, o conjunto dos tempos de fim F_g, e as capacidades disponíveis $RD_k(t)$, para cada tempo de fim t são calculados. De seguida é seleccionada uma actividade j do conjunto das actividades disponíveis D_g.

O tempo de fim de cada actividade j é calculado considerando o tempo de fim mais cedo, FMC_j, das suas actividades predecessoras e a disponibilidade de recursos.

O pseudo-código do procedimento SGS Série pode ser descrito da seguinte forma:

fig. 2.4 Pseudo-código do procedimento SGS Série.

Inicialização: $F_0=0$, $S_0=\{0\}$,

Para $g=1$ **até** n

{

Calcular D_g, F_g, $RD_k(t)$ (para todos os recursos k e todos os tempos de fim em CF_g)

Seleccionar uma actividade $j \in D_g$

Determinar o tempo de fim mais cedo da actividade j ignorando a disponibilidade de recursos

$FMC_j = Max_{l \in P_j}\{F_l\} + d_j$

Determinar o tempo de fim da actividade j considerando a disponibilidade de recursos

$$F_j = Min\{t \in \left[FMC_j - p_j, \infty\right] \cap CF_g \mid r_{j,k} \le RD_k(\tau),$$

$$k \in K, \tau \in \left[t, t+p_j\right[\cap CF_g\} + d_j$$

Actualizar o conjunto das actividades já sequenciadas

$S_g = S_{g-1} \cup \{j\}$

}

O *makespan* do projecto será dado pelo maior dos tempos de fim das actividades predecessoras da actividade $n+1$, isto é, $F_{n+1}=Max_{l \in P_{n+1}}\{F_l\}$.

Para ilustrar a aplicação do método série ao problema RCPSP, descreve-se na **fig. 2.2**, um exemplo com dois recursos, $K=2$, e seis actividades, $n=6$. Para este problema foi obtida uma duração total de 15 unidades de tempo.

A **tab. 2.1** ilustra a aplicação do SGS Série ao exemplo da **fig. 2.2**.

tab. 2.1 Aplicação do sgs Série ao exemplo da fig. 2.2.

g	1	2	3	4	5	6
Dg	{1,2}	{2}	{3,4}	{3,6}	{3}	{5}
j	1	2	4	6	3	5

Outros trabalhos sobre o método SGS Série podem ser encontrados em Pascoe (1966), Cooper (1976, 1977), Boctor (1990), Valls *et al.* (1992) e Kolisch (1996).

2.2.4.2.1.2 SGS Paralelo

Ao método SGS Paralelo estão associados dois algoritmos propostos por Kelley (1963) e Brooks e White (1965). O esquema de geração de planos baseado no SGS paralelo, que apenas constrói planos não atrasados, utiliza o incremento de tempo. Para cada estágio g, existe um tempo de programação t_g. As actividades que já foram programadas até ao estágio g, pertencem ao conjunto C_g ou ao conjunto A_g. O conjunto $C_g = \{j \in J \mid F_j \leq t_g\}$ representa todas as actividades que foram concluídas até ao instante t_g. O conjunto das actividades activas (em curso) são representadas por A_g.

As actividades que estão disponíveis para serem seleccionadas e programadas estão representadas no conjunto de actividades D_g. A capacidade disponível de cada recurso k no instante t_g é dada por $RD_k(t_g)$.

O pseudo-código do procedimento SGS Paralelo é apresentado na **fig. 2.5**.

fig. 2.5 Pseudo-código do procedimento SGS Paralelo.

Inicialização: $g=0, t_g=0, A_0=\{0\}, C_0=\{0\}, RD_k(0)=R_k(k \in K)$

Enquanto $|A_g U C_g| \leq n$ **Repetir**
{
  Passo 1
    $g = g + 1$
    $t_g = Min_{j \in Ag}\{F_j\}$
    Calcular $C_g, A_g, RD_k(t_g), D_g$
  Passo 2
    Enquanto $D_g \neq \{\}$ **Repetir**
    {
      Seleccionar $j \in D_g$
      $F_j = t_g + d_j$
      Calcular $A_g, RD_k(tg), D_g$
    }
}

Na inicialização do SGS Paralelo considera-se o tempo de programação t_g igual a zero, atribui-se a actividade de início aos conjuntos C_0 e A_0 e calcula-se a capa-

cidade disponível para cada recurso k. Cada estágio, consiste nos seguintes dois passos:

1 Determinação do próximo tempo de programação t_g e dos conjuntos associados C_g, A_g, D_g e $RD_k(t_g)$.
2 Programação das actividades disponíveis para as quais existem recursos k necessários. De observar que este esquema de geração de planos pode ter menos de n estágios, mas tem exactamente n actividades a seleccionar e programar.

A **tab. 2.2** ilustra a aplicação do SGS paralelo ao exemplo da **fig. 2.2.** A **fig. 2.1** representa a respectiva utilização de recursos.

tab. 2.2 Aplicação do sgs Paralelo ao exemplo da fig. 2.2.

g	1	1	2	3	3	4	5	6
t_g	0	0	3	4	4	6	10	11
D_g	{1,2}	{2}	{ }	{3,4}	{3}	{6}	{5,6}	{6}
j	1	2		4	3		5	6

Trabalhos experimentais utilizando o método SGS Paralelo são reportados por Pascoe (1966), Patterson (1973, 1976), Davis e Patterson (1975), Thesen (1976), Whitehouse e Brown (1979), Elsayed (1982), Alvarez-Valdes e Tamarit (1989), Ulusoy e Özdamar (1989), Boctor (1990) e Valls *et al.* (1992). Implementações sobre o método SGS Paralelo podem ser vistas, por exemplo, em Arora e Sachdeva (1989) e Kolisch (1996).

2.2.4.2.1.3 Programação para trás

Os esquemas de geração de planos utilizam a direcção do tempo em ordem crescente. Alternativamente as actividades de um problema RCPSP podem ser programadas em direcção inversa, i.e., começando com a actividade fictícia final, gradualmente calendarizam-se as actividades, terminando na actividade fictícia inicial. Este tipo de programação pode ser conseguido invertendo as relações de precedência e aplicando em sentido inverso o esquema série ou paralelo.

O problema principal deste esquema reside no facto de ser desconhecida a duração do plano a obter. Para se ultrapassar este problema arbitra-se um valor para a duração do projecto e após a obtenção do plano faz-se um ajustamento temporal, considerando que a actividade fictícia de início tem no instante inicial o valor zero.

.2.4.2.1.4 Programação bi-direccional

Os esquemas de geração de planos podem ser combinados utilizando programação para a frente e para trás por forma a se obter um esquema de geração de planos bi-direccional.

.2.4.2.2 Regras de prioridade

Nesta secção apresentam-se algumas das regras de prioridade utilizadas com frequência na literatura. No estudo computacional de Klein (2000) um total de 73 regras de prioridade foram avaliadas. Estas regras de prioridade podem ser classificadas em cinco grandes categorias, baseadas no tipo de informação que é necessária para calcular a lista de prioridades (Lawrence, 1985).

As cinco categorias propostas por Klein (2000), são:

Regras de prioridade baseadas nas actividades consideram apenas a informação que está relacionada com a própria actividade, ignorando qualquer outro tipo de informação. Seguidamente apresentam-se algumas regras de prioridade baseadas nas informações das actividades:

SPT (*Shortest Processing Time*) é seleccionada a actividade com menor tempo de processamento das actividades na lista de prioridades;
LPT (*Longest Processing Time*) é seleccionada a actividade com maior tempo de processamento das actividades na lista de prioridades;

Normalmente nesta categoria são utilizadas regras de prioridade adaptadas do planeamento de operações em ambiente *job shop*.

Regras de prioridade baseadas na rede do projecto consideram apenas a informação que está contida na rede, (dependências entre actividades). Seguidamente apresentam-se algumas regras de prioridade baseadas nas dependências das actividades:

MIS (*Most Immediate Successors*) é seleccionada a actividade que tem mais sucessoras imediatas;
MTS (*Most Total Successors*) é seleccionada a actividade que tem mais sucessoras;
LNRJ (*Least Non-related Jobs*) é seleccionada a actividade que tem o menor número de relações de precedência com as restantes;

GRPW (*Greatest Rank Positional Weight*) é seleccionada a actividade com a maior soma resultante da adição da sua duração e da duração das sucessoras imediatas;
GRPW* (*Greatest Rank Positional Weight*) é seleccionada a actividade com a maior soma resultante da adição da sua duração e da duração da totalidade das sucessoras.

Regras de prioridade baseadas no caminho crítico consideram a informação obtida no cálculo do caminho crítico. Algumas regras de prioridade baseadas no caminho crítico são:

EST (*Earliest Start Time*)
Programação para a frente: escolha de prioridade por EST
Programação para trás: igual a programação para a frente baseado na regra LFT;
EFT (*Earliest Finish Time*)
Programação para a frente: escolha de prioridade por EFT
Programação para trás: igual a programação para a frente baseado na regra LST;
LST (*Latest Start Time*)
Programação para a frente: escolha de prioridade por LST
Programação para trás: igual a programação para a frente baseado na regra EFT;
LFT (*Latest Finish Time*)
Programação para a frente: escolha de prioridade por LFT
Programação para trás: igual a programação para a frente baseado na regra EST.

Regras de prioridade baseadas nos recursos consideram apenas a informação relativa à utilização dos recursos pelas actividades. Seguidamente apresentam-se algumas regras de prioridade baseadas nos recursos:

GRD (*Greatest Resource Demand*) é seleccionada a actividade que tem maior necessidade de recursos. A necessidade de recursos de uma actividade é igual ao produto da sua duração pela soma das necessidades de recursos;
GCUMRD (*Greatest Cumulative Resource Demand*) é seleccionada a actividade que tem maior necessidade acumulada de recursos. A necessidade acumulada de recursos de uma actividade é igual à soma da necessidade de recursos da actividade, acrescida da necessidade de recursos de todas as suas actividades sucessoras imediatas.

Regras de prioridade compostas podem ser obtidas como uma soma pesada dos valores das prioridades obtidas pelas regras descritas anteriormente. Seguidamente apresentam-se algumas regras de prioridade compostas:

wrup (*Weighted Resource Utilisation and Precedence*) esta prioridade é igual à soma pesada do número de sucessoras da actividade e da utilização média dos recursos (necessidade de recursos / capacidade de recursos), para todos os tipos de recursos. Saliente-se o trabalho de Ulusoy e Ozdamar (1989) que propôs 0.7 para o primeiro termo e 0.3 para o segundo termo;

IRSM (*Improved Resource Scheduling Method*) regra de prioridade dinâmica que é descrita por Kolisch (1996a) e que melhora o método de programação de recursos de Brand *et al.* (1964). Esta regra calcula a extensão da duração do projecto caso uma actividade elegível seja programada. Esta regra programa a actividade que dá origem à menor extensão da duração do projecto de entre todas as actividades elegíveis;

WCS (*Worst Case Slack*) regra de prioridade dinâmica descrita por Kolisch (1996 a). Para cada actividade elegível é calculada a folga que resta. A actividade com a menor folga é seleccionada para ser programada.

Vários estudos que comparam diferentes regras de prioridade foram desenvolvidos por Davis e Patterson (1975), Alvarez-Valdés e Tamarit (1989), Boctor (1990), Valls *et al.* (1992) e Kolisch (1995, 1996 a).

2.4.2.3 Métodos multi-passagem

A utilização de mais do que um esquema de geração de planos, com uma ou mais direcções de programação e uma ou mais regras de prioridade, permite construir métodos de multi-passagem.

Existem várias formas de construção de métodos de multi-passagem. As mais comuns são descritas de seguida:

Métodos multi-programação estes métodos são baseados nos dois esquemas de geração de planos (série e paralelo) e nas três direcções de programação (para a frente, para trás e bidireccional), totalizando seis combinações possíveis. Utilizando apenas uma regra de prioridade, todas as seis combinações podem ser testadas e escolhida a que obtém a melhor solução. Alternativamente, utilizando uma regra de prioridade dinâmica (por exemplo, MSLKD ou WCS), pode-se iterativamente programar um projecto alternando as seis combinações, utilizando valores de prioridade que são baseados no plano obtido da iteração anterior. Salientam-se nesta área os trabalhos nesta área de Li e Willis (1992), Ozdamar e Ulusoy (1996 a,b) e Tormos e Lova (2001);

Métodos de múltiplas regras de prioridade estes métodos utilizam um esquema de geração de planos e uma direcção de programação com várias regras

de prioridade. Salientam-se os trabalhos nesta área de Boctor (1990), Ulusoy e Ozdamar (1989) e Thomas e Salhi (1997);

Métodos probabilísticos estes métodos utilizam um esquema de geração de planos (normalmente com programação para a frente), e uma regra de prioridade. Por forma a evitar a geração de planos idênticos, a selecção da próxima actividade a ser planeada é determinada estocasticamente, baseada nos valores de prioridade para as diferentes actividades. As diversas formas de cálculo das probabilidades de cada actividade são classificadas como *random sampling, biased random sampling* e *regret based biased random sampling.* Vários têm sido os trabalhos nesta área, salientando-se Cooper (1976), Alvarez-Valdés e Tamarit (1989), Schirmer e Riesenberg (1997), Drexl (1991), Kolisch (1995), Kolisch e Drexl (1996) e Schirmer (1998).

2.2.4.3 Metaheurísticas

Nas duas últimas décadas um número crescente de meta-heurísticas tem vindo a ser proposto por vários autores para a resolução de problemas de planeamento de projectos.

Nas secções seguintes apresentam-se resumidamente as abordagens das meta-heurísticas: *tabu search*, *simulated annealing* e algoritmos genéticos.

2.2.4.3.1 *Tabu search*

Glover (1989, 1990) foi o percursor da abordagem *tabu search*. Esta abordagem consiste em pesquisar todas as soluções na vizinhança de uma solução obtida através de uma heurística construtiva. Todas as soluções viáveis na vizinhança de uma solução são avaliadas e a que tiver melhor valor é escolhida. No caso de não se encontrar uma melhor solução, este processo de pesquisar soluções na vizinhança de uma solução pode continuar indefinidamente. Esta característica do *tabu search* torna possível que um conjunto de soluções crie ciclos, i.e., que um subconjunto de soluções seja pesquisado repetidamente. Para evitar a criação de ciclos, o *tabu search* utiliza um mecanismo que proíbe determinadas movimentações consideradas como *tabu*.

Pinson *et al.* (1994) apresentam três heurísticas *tabu search* que são baseadas num esquema de representação tipo lista de prioridades e utilizam um esquema de geração de planos do tipo série. As três heurísticas diferem no operador utilizado: o primeiro é baseado no operador de troca par, o segundo é baseado no operador de troca geral e o terceiro é baseado no operador de mudança.

Lee e Kim (1996) apresentam uma heurística *tabu search* que é baseada no es-

quema de representação do tipo chaves aleatórias e utiliza o esquema de geração de planos do tipo paralelo. O operador utilizado é do tipo de troca par.

Baar *et al.* (1998) apresentam duas heurísticas *tabu search*. Uma primeira é baseada num esquema de representação tipo lista de prioridades e utiliza um esquema de geração de planos do tipo série com três diferentes operadores de mudança. A segunda heurística é baseada num esquema de representação do tipo esquema de plano baseado em Brucker *et al.* (1998) e utiliza um esquema de geração de planos, baseado no trabalho de Baar *et al.* (1998), com quatro diferentes operadores, que realizam transformações de relações de flexibilidade em relações de paralelismo e vice-versa.

Neste tipo de abordagem salientam-se ainda os trabalhos de Thomas e Salhi (1998) e Nonobe e Ibaraki (2001).

2.4.3.2 *Simulated annealing*

O conceito da abordagem *simulated annealing* foi proposto por Metropolis *et al.* (1953). Metropolis *et al.* descrevem um algoritmo para simular o arrefecimento de material numa imersão quente, num processo conhecido como *annealing*. O algoritmo proposto simula uma mudança de estado energético quando o sistema global é sujeito a um processo de arrefecimento, até convergir para um estado estacionário.

A abordagem *simulated annealing* começa com uma solução inicial obtida por uma heurística construtiva. Uma nova solução é criada na vizinhança desta solução e é aceite se é melhor que a solução inicial ou corrente. Se uma nova solução não melhora a solução corrente pode ser aceite com uma probabilidade que depende da temperatura. Esta temperatura começa com um valor elevado e vai diminuindo durante o *simulated annealing* por forma a diminuir a probabilidade de aceitação de soluções de menor valor. Este processo é repetido até ser executado um período de tempo, serem criadas um determinado número de soluções ou nenhuma solução ter sido aceite num determinado período de tempo de execução.

Sampson e Weiss (1993) apresentam um algoritmo que é um variante do *simulated annealing* sendo baseado no esquema de representação de mudança vectorial e utiliza um operador de troca.

Boctor (1996b) apresenta uma adaptação do *simulated annealing* que é baseada num esquema de representação do tipo lista de prioridades e utiliza um esquema de geração de planos do tipo série. O operador utilizado é um operador de mudança.

Lee e Kim (1996) apresentam um algoritmo *simulated annealing* que é baseado num esquema de representação do tipo chaves aleatórias e utilizam um esquema de geração de planos do tipo paralelo, com um operador do tipo troca par.

Bouleimen e Lecocq (1998, 2003) apresentam um algoritmo *simulated annealing* que é baseado num esquema de representação do tipo lista de prioridades e utiliza um esquema de geração de planos do tipo série.

2.2.4.3.3 Algoritmos genéticos

Uma terceira abordagem meta-heurística, que se designa por algoritmos genéticos, é baseada nos mecanismos da evolução biológica e genética natural. Os algoritmos genéticos foram introduzidos por Holland em 1975 que tenta implementar a ideia da sobrevivência dos mais aptos no campo da optimização combinatória.

Um algoritmo genético começa com uma população de n soluções que foram geradas por uma heurística construtiva. A partir das n soluções, são geradas novas soluções através da combinação de duas soluções utilizando um tipo de operador binário e de um operador de mutação. O operador de mutação é um operador unário que é aplicado com uma baixa probabilidade a algumas soluções. A nova população consiste das n melhores soluções geradas e/ou existentes e o algoritmo continua com a população corrente. Este processo é repetido até determinado número de populações terem sido geradas e avaliadas.

Leon e BalaKrishnan (1995) apresentam uma abordagem de um algoritmo genético baseado num esquema de representação tipo chaves aleatórias e utiliza um esquema de geração de planos do tipo paralelo modificado. São utilizados os operadores de troca de Storer *et al.* (1992) e o de cruzamento de ponto único.

Hartmann (1998) apresenta três abordagens de algoritmos genéticos, uma baseada num esquema de representação do tipo lista de prioridades, uma segunda baseada num esquema de representação do tipo regra de prioridade e uma terceira utiliza um esquema de representação do tipo chaves aleatórias. Todas as abordagens utilizam um esquema de geração de planos do tipo série e um operador de cruzamento de duplo ponto.

Kohlmorgen *et al.* (1999) apresenta uma abordagem de um algoritmo genético baseado num esquema de representação tipo chaves aleatórias, utiliza um esquema de geração de planos do tipo série e um operador de cruzamento de duplo ponto.

De salientar outros trabalhos nomeadamente Leon e Ramamoorthy (1995) e Hartmann (2002).

2.3 A Revisão da literatura – RCMPSP

As abordagens referidas na secção 2.2 tratam o problema do planeamento de um único projecto. O problema RCMPSP é consideravelmente mais complexo do que o problema RCPSP tratado na secção anterior. Por consequência, o trabalho dos investigadores tem sido orientado, na maioria dos casos, para métodos aproximados.

2.3.1 Métodos exactos

2.3.1.1 Modelos de programação matemática

Pritsker *et al.* (1969) apresentam um dos trabalhos pioneiros no problema RCMPSP, na qual propõem uma abordagem baseada em programação inteira.

Mohanty e Siddiq (1989) estudaram o problema RCMPSP com datas de entrega. O estudo é executado usando programação inteira e simulação. O modelo de programação inteira é usado para gerar os planos. O modelo de simulação é empregado para testar algumas regras heurísticas com diferentes cenários em termos de carga de recursos. Finalmente o modelo compara os planos gerados por ambas as alternativas e selecciona o melhor plano.

Deckro *et al.* (1991) apresentam um algoritmo com uma abordagem de decomposição baseada em programação inteira para problemas de grande dimensão.

Leachman (1993) apresenta um algoritmo para solucionar o problema do RCMPSP. Este algoritmo é baseado em programação linear tendo como objectivo minimizar o custo total dos atrasos.

Vercellis (1994) apresenta uma abordagem que utiliza decomposição lagrangeana.

Shankar e Nagi (1996) apresentam um algoritmo que considera as fases de planeamento e de programação. A fase de planeamento utiliza programação linear e a fase de programação utiliza *simulated annealing*.

Wiley *et al.* (1998) desenvolveram um algoritmo utilizando a estrutura da divisão do trabalho (WBS) e a decomposição de Dantzig-Wolfe. O procedimento de Dantzig-Wolfe permite a geração de soluções admissíveis. O algoritmo desenvolvido permite ao utilizador efectuar análise de sensibilidade, programação paramétrica e definir condições e limites.

2.3.2 Métodos aproximados

2.3.2.1 Regras de prioridade

Fendley (1968) foi um dos pioneiros na programação de múltiplos projectos. Analisou o comportamento de várias heurísticas com 3 e 5 projectos, com três medidas de desempenho: atrasos em relação às datas de entrega, nível de utilização de recursos e nível dos em curso de fabrico. A principal conclusão do seu trabalho foi que a regra de prioridade MINSLK (folga mínima) obtém os melhores resultados para as três medidas.

Kurtulus e Davis (1982) utilizaram problemas em que os projectos tinham entre 34 e 63 actividades e cada actividade necessitava entre 2 e 6 unidades por recurso.

Neste estudo apresentaram seis novas regras de prioridade. A medida utilizada para desempenho foi a média dos atrasos dos projectos.

Kurtulus e Narula (1985) utilizaram problemas com 3 projectos, em que cada projecto tinha entre 24 e 33 actividades para pequenos problemas e entre 50 e 60 actividades para grandes problemas. Os autores utilizaram as seis regras de prioridade de Kurtulus e Davis (1982) e quatro novas regras de prioridade baseadas em penalidades. A conclusão principal do trabalho foi a de que a nova regra de prioridade *maximum penalty* obteve os melhores resultados para a medida soma dos pesos dos atrasos.

Dumond e Mabert (1988) apresentam cinco regras heurísticas e quatro estratégias para o cumprimento das datas de entrega dos projectos. O objectivo é seleccionar a melhor regra de prioridade que minimiza os atrasos.

Um algoritmo para controlar projectos foi desenvolvido por Tsubakitani e Deckro (1990) para o problema RCMPSP. O algoritmo implementou a regra de prioridade SASP (*Shortest Activity from the Shortest Project*) para planeamento de mais de 50 projectos, os quais podiam ter mais do que 100 actividades cada. O algoritmo considera também a possibilidade do utilizador actualizar os projectos em fase de execução.

Lawrence e Morton (1993) estudaram o problema das datas de entrega e desenvolveram um algoritmo que obtém um bom desempenho em termos de minimização da média dos atrasos dos projectos.

Villeneuve *et al.* (1997) apresentam um algoritmo de ordenação dinâmica para o problema RCMPSP aplicado à indústria de reconstrução. Este algoritmo utiliza tempos estocásticos para a duração das actividades e tem como objectivo o cumprimento das datas de entrega dos projectos.

2.3.2.2 Métodos multi-passagem

Lova *et al.* (2000) apresentam uma heurística multi-passagem para melhorar a afectação de recursos. Esta heurística compreende duas fases. A primeira fase obtém um plano admissível através de uma regra de prioridade, utilizando programação bi-direccional. A segunda fase tenta melhorar a afectação de recursos através de sucessivas passagens bi-direccionais, sem aumentar a duração dos projectos obtida na primeira fase.

2.4 A Revisão da literatura – *Job Shop*

O problema RCPSP é uma generalização do problema clássico JSP, Kolisch (1995). Apesar do problema JSP ser um caso particular do problema RCPSP, tem sido ob-

jecto de estudo considerável pelos investigadores. O problema JSP é NP-difícil e é também um dos mais difíceis problemas de programação, Brucker *et al.* (1992).

A história do problema JSP remonta a mais de 30 anos, e é também a história de um conhecido problema de optimização da programação de 10 ordens de fabrico em 10 máquinas, introduzido por Fisher e Thompson em 1963. Este problema originou uma grande competição entre os investigadores durante 25 anos de forma a encontrar o valor óptimo para o *makespan*.

O processo de programação de operações é de tal forma complexo que só recentemente foi possível provar a optimalidade das soluções para certos problemas. Carlier e Pinson (1989) desenvolveram um método eficiente de *branch and bound* utilizando técnicas melhoradas de *bounding* bem como uma nova regra heurística de selecção de *branch*. Usando este algoritmo, eles foram os primeiros a provar a optimalidade da solução (*makespan* de 930) para o problema de 10 ordens de fabrico em 10 máquinas proposto por Fisher e Thompson. Este método foi aperfeiçoado por Applegate e Cook (1991) tendo introduzido limites inferiores (*lower bounds*) mais aproximados e uma regra heurística de selecção de branch mais eficiente reduzindo significativamente o tempo de processamento do algoritmo.

Foi também investido bastante esforço no teste empírico de várias regras de prioridade. Giffler aplicou uma regra de prioridade baseada em probabilidades para a selecção de operações de forma a construir uma boa solução.

Recentemente, novas técnicas heurísticas foram introduzidas para resolver estes problemas. Uma das melhores é a *Shifting Bottleneck*, tendo tido bom desempenho em muitos problemas e sendo também utilizada em conjunto com outros métodos heurísticos.

Os métodos heurísticos mais promissores são metaheurísticas, das quais se destacam os algoritmos genéticos, *tabu search* e *simulated annealing*.

Excelentes trabalhos de análise, abordando os diferentes métodos para a resolução do problema do *job shop*, têm sido propostos por vários autores, salientando-se Gere (1966), Mellor (1966), Panwalker (1977), Baker *et al.* (1981) e mais recentemente Pinson (1995), Vaessens *et al.* (1996), Cheng *et al.* (1999) e Jain e Meeran (1999).

Nas secções seguintes são apresentados, resumidamente, alguns dos algoritmos utilizados na resolução de problemas de programação de operações em ambientes do tipo *job shop*, classificando-os em 4 categorias: representação gráfica, métodos exactos, métodos aproximados e outras abordagens.

4.1 Representação gráfica

Roy e Sussmann (1964) propuseram uma representação gráfica para o problema do JSP que designaram por grafo disjuntivo, representado por $G=(V, C \cup D)$.

O conjunto dos vértices, V, é formado pelo conjunto de todas as operações, $J=\{0, 1, \ldots, n, n+1\}$, e por duas operações fictícias, 0 e $n+1$, que no grafo representam uma operação inicial para todas as ordens de fabrico e uma operação final para todas as ordens de fabrico, $V=\{0, 1, \ldots, n, n+1\}$. A cada vértice j é associado um tempo de processamento d_j, da j-ésima operação. Os vértices 0 e $n+1$ têm duração nula.

O conjunto dos arcos é formado por um conjunto C de arcos conjuntivos (arcos orientados) e por um conjunto D de arcos disjuntivos (arcos não-orientados). No conjunto C existe um arco por cada par ordenado de operações (i, j), relacionadas pelas restrições de precedência i p j, orientado de i para j. Estes arcos definem a ordem de processamento das operações que pertencem à mesma ordem de fabrico.

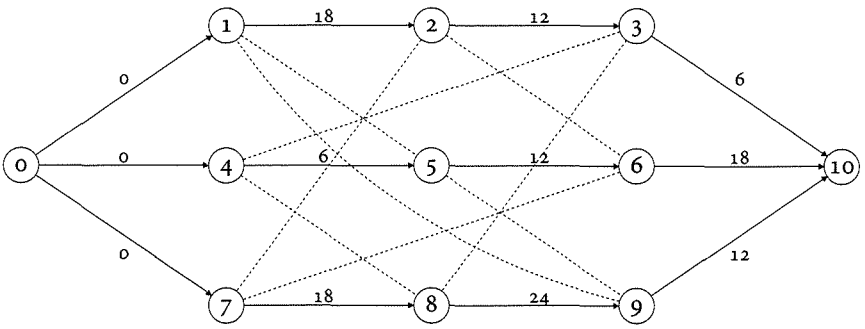
Para cada máquina m de $M=\{1, \ldots, m\}$, define-se o conjunto D_m formado por todos os pares de operações que são processados nessa máquina. Para cada par de operações $(i, j) \in D_m$ existe um arco sem sentido definido (operações não se podem sobrepor). Diz-se que as operações i e j definem um arco disjuntivo. Assim os arcos sem sentidos definidos (disjuntivos) representam o facto de que cada máquina só pode processar uma operação em cada momento.

A **fig. 2.6** ilustra o grafo disjuntivo aplicado à representação do problema de três ordens de fabrico por três máquinas, conforme dados da **tab. 2.3**.

tab. 2.3 Ordem e tempos de processamento das operações.

Ordens de Fabrico	1ª Operação		2ª Operação		3ª Operação	
	Máquina	Tempo	Máquina	Tempo	Máquina	Tempo
1	1:1	18	2:2	12	3:3	6
2	4:3	6	5:2	2	6:2	18
3	7:2	18	8:3	4	9:1	12

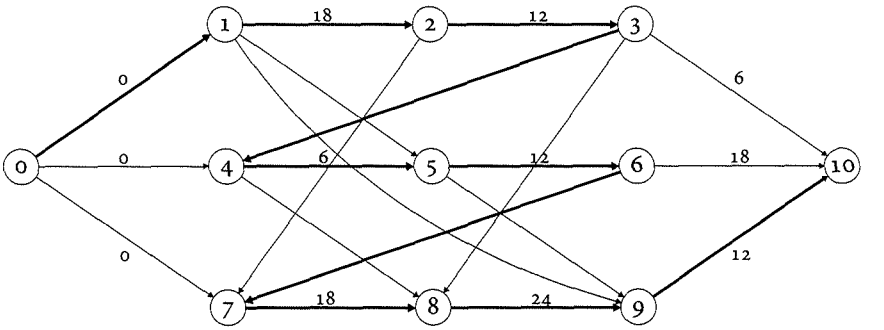
fig. 2.6 Exemplo de grafo disjuntivo.



O princípio de programação do grafo disjuntivo consiste na atribuição de um sentido aos arcos disjuntivos, isto é, definição uma ordem de processamento entre todas as operações que são processadas pela mesma máquina. A selecção dos sentidos dos arcos deve ser realizada de tal forma que o grafo resultante G, seja acíclico (não existam conflitos de precedência entre as operações) e que o comprimento do caminho mais longo (também designado por caminho crítico) entre 0 e $n+1$ seja mínimo. O *makespan* de um plano é igual ao comprimento do caminho crítico. Qualquer arco (i,j) pertencente ao caminho crítico é designado por arco crítico. Se i e j são operações de diferentes ordens de fabrico, então (i, j) é chamado arco disjuntivo crítico, caso contrário designa-se por arco conjuntivo crítico.

O comprimento do caminho mais longo, também designado por caminho crítico, entre os vértices 0 e $n+1$, do grafo acíclico G, é igual ao valor de C_{max}, o maior tempo de conclusão de processamento de todas as ordens de fabrico. Na **fig. 2.7** apresenta-se o caminho crítico relativo ao problema da **tab. 2.3**.

fig. 2.7 Grafo disjuntivo com caminho critico.



No exemplo apresentado o caminho crítico tem o valor de 126 e o plano passa nos vértices 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10.

2.4.2 Métodos exactos

Os métodos exactos têm um grande interesse teórico, na medida em que permitem obter sugestões para novos métodos e regras. Contudo o seu interesse prático é pequeno, dado os pressupostos em que se baseiam, Schroeder (1989).

O problema de calendarização de n ordens de fabrico em m máquinas foi resolvido para $m=1,2$ e para valores arbitrários de n. Para alguns casos particulares de $m=3$, com restrições e qualquer número de ordens de fabrico n.

Algoritmos eficientes não foram ainda desenvolvidos para $m \geq 4$.

2.4.2.1 Algoritmo de Smith

Smith (1956) desenvolveu um algoritmo que minimiza o tempo médio de fluxo de n ordens de fabrico, com a mesma data de libertação, numa só máquina.

2.4.2.2 Algoritmo de Moore

Este algoritmo minimiza o número total de ordens de fabrico atrasadas para o caso em que existem n ordens de fabrico, com a mesma data de libertação, numa só máquina. O algoritmo é constituído pelos seguintes passos, Buffa e Sarin (1987):

Passo 1 Colocar as ordens de fabrico por ordem crescente das suas datas de entrega;

Passo 2 Se esta sequência produz zero ordens atrasadas então a sequência é óptima, se existirem ordens de fabrico removidas calendarizá-las no fim da sequência por qualquer ordem, e o processo pára, senão identificar a primeira ordem de fabrico atrasada e passo 3;

Passo 3 Identificar a ordem de fabrico com maior tempo de processamento entre a primeira atrasada e todas as que terminaram a tempo. Remover essa ordem e recalcular a sequência. Retornar ao passo 2.

Algumas modificações e extensões têm sido realizadas neste algoritmo.

Kise *et al.* (1978), modificou este algoritmo de forma a resolver problemas com tempos de libertação diferentes, mas com a restrição $d_i \leq d_j \Rightarrow r_i \leq r_j$, isto é, os tempos de libertação, r_i e r_j, devem estar na mesma sequência que as datas de entrega, d_i e d_j.

Sidney (1973), modificou o algoritmo de Moore numa direcção diferente. O seu algoritmo admite que certas ordens de fabrico possam ser especificadas como não atrasadas, isto é, a sequência deve assegurar que determinadas ordens de fabrico estejam completadas até às suas datas de entrega.

2.4.2.3 Algoritmo de Sidney

Sidney (1977) considera o problema da calendarização numa só máquina de n ordens de fabrico, com datas de libertação e datas de entrega diferentes, e que podem incorrer em penalidades por motivo de atraso ou de avanço. O algoritmo minimiza a maior penalidade.

4.2.4 Algoritmo de Lawer

Aplica-se a *n* ordens de fabrico e uma máquina, com restrições de precedência entre as ordens de fabrico a produzir.

Lawer (1973) desenvolveu um algoritmo que permite minimizar o maior custo do processamento das *n* ordens de fabrico numa só máquina quando existem restrições entre as ordens de fabrico. O custo é função da data de conclusão.

4.2.5 Algoritmo de Jonhson

Este algoritmo foi desenvolvido por Jonhson (1954) e minimiza o tempo total de fabrico para o caso da programação de *n* ordens de fabrico, com datas de libertação iguais, em duas máquinas.

4.2.6 Algoritmo de Szwarc

Este algoritmo foi desenvolvido por Szwarc (1977) e minimiza o tempo total de fabrico para o caso da programação de *n* ordens de fabrico, com datas de libertação idênticas, em três máquinas.

4.2.7 Modelos de programação matemática

Vários algoritmos de programação matemática foram desenvolvidos ao longo do tempo, sendo de salientar alguma literatura tal como Wagner (1959), Bowman (1959), Dantzig (1960), Manne (1960), Agin (1966), Lawler and Wood (1966), Conway *et al.* (1967), Baker (1974), Macmahon *et al.* (1975), Rinnooy Kan (1976), Lenstra (1977), French (1982), Croce *et al.* (1993), Hoitomt *et al.* (1993), Kruger *et al.* (1995) e Williamson *et al.* (1997).

Os modelos mais utilizados resolvem o problema do planeamento operacional com algoritmos padrão desenvolvidos para resolver problemas matemáticos gerais.

Embora os modelos da programação matemática pareçam ser uma abordagem promissora, na prática não o são. Estes modelos são pouco utilizáveis uma vez que se aplicam somente a problemas de dimensões reduzidas quando comparáveis com os problema reais.

Várias outras formulações no âmbito da programação matemática têm sido propostas, nomeadamente programação dinâmica. Este tipo de formulação teve a sua origem no trabalho de Bellman em 1957. Held e Karp (1962) foram os per-

cursores da aplicação dos modelos de programação dinâmica aos problemas de programação. Outros autores se seguiram tal como Lawer e Moore (1969), Conway *et al.* (1967), Baker (1974), Corwin e Esogbue (1974), Baker e Schrage's (1978), Smeds (1980) e French (1982).

A vantagem dos métodos de programação dinâmica resulta da redução do esforço computacional necessário, em relação à enumeração completa de hipóteses, para resolver problemas de programação.

A **tab. 2.4** ilustra o número de operações necessárias para a programação dinâmica e enumeração, para n ordens de fabrico e 1 máquina.

tab. 2.4 Comparação do esforço computacional necessário para resolver um problema por enumeração completa e programação dinâmica.

	Número de operações requeridas por:	
Número de ordens de fabrico	Enumeração completa	Programação dinâmica
4	645	645
10	2.286×10^{8}	33789
20	2.992×10^{20}	6.396×10^{7}
40	1.983×10^{50}	1.352×10^{14}

Aumentando o número de ordens de fabrico e de máquinas o número de operações a efectuar é muito elevado tornando impraticável a utilização da programação dinâmica a problemas de grande dimensão.

2.4.2.8 Modelos baseados em *branch and bound*

Alguns dos trabalhos que abordam o problema do planeamento operacional por algoritmos utilizando a técnica *branch and bound* são, por exemplo, Balas (1969), Florian *et al.* (1971), McMahon e Florian (1975), Carlier (1982), Carlier (1987), Carlier e Pinson (1989, 1990), Applegate e Cook (1991), Brucker *et al.* (1994), Perregaard e Clausen (1995), Boyd e Burlingame (1996) e Martin (1996).

2.4.3 Métodos aproximados

Os problemas de programação são NP-difícil, e como consequência é de supor que não existam algoritmos capazes de fornecer uma solução óptima em tempo útil.

Os problemas reais têm no entanto que ser resolvidos em tempo útil mesmo que a solução obtida não seja a óptima. Os métodos aproximados têm por objectivo obter planos de boa qualidade em tempo útil.

Várias contribuições sobre algoritmos aproximados têm sido dadas ao longo do tempo nomeadamente por Giffler e Thompson (1960), Heller e Logemann (1962), Palmer (1965), Campbell *et al.* (1970), Holloway (1974), Gonzalez (1978), French (1982), Adams *et al.* (1988) e Gray e Hoesada (1991).

Seguidamente apresentam-se alguns métodos aproximados.

4.3.1 Algoritmo de Palmer

Este algoritmo minimiza o maior tempo de fluxo de uma ordem de fabrico para o caso em que existam n ordens de fabrico e m máquinas, num ambiente de *flow shop*.

Palmer (1965), desenvolveu uma heurística cuja ideia é obter um índice de variação resultante das ordens de fabrico tenderem a evoluir de tempos de processamento pequenos para grandes tempos de processamento, à medida que passam de máquina para máquina. Assim, quando as ordens de fabrico são calendarizadas em ordem decrescente do seu índice de variação, será de esperar encontrar uma sequência próxima da óptima.

O índice de variação para a ordem de fabrico i é:

$$S_i = -\sum_j [m-(2j-1)]\, p_{ij} \qquad j = 1,...,m$$

sendo p_{ij} o tempo de processamento da ordem de fabrico i na máquina j.

Dannenbring (1977), aplicou a heurística de Palmer a 1280 problemas de pequena dimensão, $m \leq 10$, $n \leq 6$. A heurística de Palmer obteve o óptimo para 30% dos casos.

4.3.2 Algoritmo de Holloway e Nelson

Aplica-se a n ordens de fabrico e m máquinas. O objectivo é gerar sequências para obter soluções satisfatórias para quatro critérios: maior atraso, variância dos atrasos positivos, média dos atrasos positivos e número de ordens terminadas a tempo.

Esta heurística resulta da combinação da regra SPT e da dimensão da fila de espera, Q, em cada centro, Holloway e Nelson (1974). Para a calendarização de cada máquina utiliza-se a regra SPT, mas se a fila da próxima máquina tem mais que um determinado número de operações, tenta-se uma nova operação com a regra SPT.

Esta heurística é apresentada em duas formas: produzindo planos activos, que resultam de uma máquina esperar que cheguem outras operações para processa-

mento; e planos não-atrasados, que resultam de uma máquina iniciar o processamento com uma das operações disponíveis.

2.4.3.3 Algoritmo de Giffler e Thompson

O algoritmo de Giffler e Thompson (1960) aplica-se a n ordens de fabrico e m máquinas. Este algoritmo gera planos activos.

No algoritmo calendariza-se uma operação de cada vez. Uma operação pode ser calendarizada se dentro da sua ordem de fabrico todas as operações que a precedem já foram calendarizadas. Se existirem nm operações, o algoritmo terá nm iterações ou estados.

No estado t temos:

- P_t plano parcial das $t-1$ operações calendarizadas,
- S_t conjunto das operações calendarizáveis no estado t, isto é, todas as operações que sucedem às que estão calendarizadas em P_t,
- σ_k tempo mais cedo que a operação σ_k em S_t pode ser iniciada,
- φ_k o tempo de fim mais cedo que a operação σ_k pode ser terminada, isto é, $\varphi_k = \sigma_k + p_k$, onde p_k é o tempo de processamento de σ_k.

Passos do algoritmo de Giffler e Thompson:

Passo 1 Seja $t=1$ com $P_1 = \varnothing$. S_1 é o conjunto de todas as operações sem predecessores, isto é, todas as primeiras operações.

Passo 2 Encontrar $\varphi^* = min\, \sigma_k$ em S_k $\{\varphi_k\}$ e a máquina M^* na qual φ^* ocorre.

Passo 3 Escolher uma operação σ_j em S_t tal que

1 necessita M^*, e

2 $\sigma_j < \varphi^*$.

Passo 4 Mover para próximo estado realizando

1 adicionar σ_j a P_t, criando P_{t+1},

2 apagando σ_j desde S_t e criando S_{t+1}, adicionando a S_t a operação que se segue a σ_j na ordem de fabrico (exceptua-se o caso de σ_j completar a ordem de fabrico),

3 incrementar t de 1.

Passo 5 Se existem operações a calendarizar ($t < nm$) então passo 2, senão parar.

O primeiro passo do algoritmo inicializa o problema. De seguida calendariza-se cada operação. O cálculo de φ^* dá o tempo de fim mais cedo de uma operação em S_k. Considera-se a máquina onde ocorre. Escolhe-se uma qualquer operação

com tempo de início anterior a φ^*. O processo é repetido até não existirem mais operações.

3.4 Algoritmo modificado de Giffler e Thompson

French (1982), apresenta uma modificação do algoritmo de Giffler e Thompson. O objectivo deste algoritmo modificado é produzir planos não-atrasados, i.e., se uma máquina estiver disponível e existir uma operação para processamento nessa máquina, inicia-se o fabrico.

French (1982) apresenta os seguintes passos do algoritmo modificado de Giffler e Thompson:

Passo 1 Seja $t=1$ com $P_1=\varnothing$. S_1 é o conjunto de todas as operações sem predecessores, isto é, todas as primeiras operações.

Passo 2 Encontrar $\sigma^* = min\ \sigma_k$ em $S_k\{\sigma_k\}$ e a máquina M^* na qual σ^* ocorre.

Passo 3 Escolher uma operação σ_j em S_t tal que

1 necessita M^*, e

2 $\sigma_j = \sigma^*$.

Passo 4 Mover para próximo estado realizando

1 adicionar σ_j a P_t, criando P_{t+1},

2 apagando σ_j desde S_t e criando S_{t+1}, adicionando a S_t a operação que se segue a σ_j na ordem de fabrico (exceptua-se o caso de σ_j completar a ordem de fabrico),

3 incrementar t de 1.

Passo 5 Se existem operações a calendarizar ($t<nm$) então passo 2, senão parar.

A modificação para alcançar o objectivo de French foi realizada nos passos 2 e 3. Escolhe-se o menor tempo de início possível, σ^*, das operações calendarizáveis, e calendariza-se a operação cujo tempo de início coincide com σ^*.

1.3.5 Algoritmo de Adams, Balas e Zawack

Este algoritmo minimiza o tempo total de fabrico para o caso em que existem n ordens de fabrico e m máquinas.

O algoritmo proposto por Adams *et al.* (1988) programa as máquinas uma por uma, sucessivamente, considerando de cada vez a máquina identificada como *bottleneck*. Sempre que uma máquina é programada, todas as sequências estabelecidas previamente são reoptimizadas. A identificação das *bottleneck* e a reoptimização são baseadas na resolução repetitiva de problemas de uma só máquina.

Seja M o conjunto de todas as máquinas do problema e $M'=\{M_1, \ldots, M_k\}$ o conjunto de todas as máquinas programadas. Desta forma podemos considerar $M \setminus M'=\{M_{k+1}, \ldots, M_m\}$ como sendo o conjunto de máquinas por programar. O pseudo-código do algoritmo *shifting bottleneck* pode ser descrito da seguinte maneira, ver **fig. 2.8**:

fig. 2.8 Pseudo-código do algoritmo *shifting bottleneck*.

$M' = \{\}$

Passo 1 Encontrar a máquina m que provoca um estrangulamento entre as máquinas do conjunto $M \setminus M'$ e fazer a sua programação de modo óptimo.

$M' = M' \cup \{m\}$

Passo 2 Re-optimizar a sequência de cada máquina crítica $k \in M'$ mantendo as sequências das outras máquinas.

Se $M' \neq M$ saltar para Passo 1 senão parar.

Para cada máquina pertencente a $M \setminus M'$ é resolvido um problema de uma única máquina baseando-se na data de disponibilidade e na data de entrega. Os valores das datas são definidos a partir do planeamento de operações nas máquinas do conjunto M'. A máquina cujo valor óptimo de *makespan* for superior é considerada a máquina crítica (máquina que provoca o estrangulamento).

Depois da programação da máquina crítica é resolvido um novo problema de máquina única, re-optimizando localmente todas as máquinas pertencentes a M'. É repetido o processo até que todas as máquinas de M estejam programadas.

Adams *et al.* (1988) testaram o seu método para 40 problemas ($10 \leq n \leq 30, 5 \leq m \leq 15$), comparando o resultado com regras de prioridade, sendo que os resultados obtidos foram tão bons ou melhores que as regras de prioridade comparadas em 96% dos casos.

2.4.3.6 Algoritmo de Gray e Hoesada

Aplica-se a n ordens de fabrico e m máquinas, tendo como objectivo encontrar um bom plano para os atrasos médios.

Gray e Hoesada (1991) apresentam um trabalho em que procuram utilizar a vantagem de cada uma das regras de prioridade SPT e FOLGA. Os resultados obtidos demonstram a importância da variação da taxa SPT/FOLGA, em função da carga em cada posto de trabalho. Quando a carga se aproxima dos 100% a melhor taxa a utilizar deverá ser 70%/30%.

.3.7 Regras de prioridade

Dentro dos métodos heurísticos existem ainda regras de prioridade. O uso destas regras de prioridade envolve o cálculo de um índice de prioridade para cada ordem de fabrico.

Uma regra de prioridade diz-se estática se as prioridades não variam com o tempo, e dinâmica caso contrário. Dizem-se locais se apenas usam a informação da ordem de fabrico a que pertencem e globais se incorporarem informação de outras máquinas ou centros de trabalho. Estas regras são práticas para a obtenção dum plano, mas, em geral, não produzem um plano óptimo.

Seguidamente apresentam-se algumas regras de prioridade usadas com mais frequência. Far-se-á uso da nomenclatura anglo-saxónica porque é a mais divulgada:

PCO (*Preferred Customer Order*) A ordem de fabrico de um cliente com preferência é processada primeiro.

SPT (*Shortest Processing Time*) Para esta regra a ordem de fabrico cuja operação sobre a máquina, tem menor tempo de processamento é selecionada. Esta regra é baseada na ideia que quando uma ordem de fabrico é terminada rápidamente, outras máquinas que estejam disponíveis recebem trabalho, resultando numa elevada taxa de fluxo e utilização. Esta regra ignora as datas de entrega ao cliente. Nesta regra, algumas ordens de fabrico podem atrasar-se, tendendo nestes casos os atrasos a serem significativos.

MINSLACK (*Folga Mínima por Ordem*) Folga é definida como o tempo disponível até à data de entrega menos o tempo de processamento restante, menos a data actual. Assim uma ordem de fabrico com folga zero tem exactamente o tempo necessário para ser realizada, se não tiver tempos de espera em filas.

SLACK/NOP (*Slack per Number of Operations*) Regra baseada na anterior, sendo a folga dividida pelo número de operações a processar. Para o caso de haver duas ordens de fabrico com a mesma folga, processa-se primeiro a que tiver maior número de operações a processar. Com efeito, essa ordem de fabrico terá mais tempos de espera e mais tempo despendido em movimentações.

SLACK/RPT (*Slack per Remaining Processing Time*) A folga é dividida pela soma dos tempos das operações por realizar.

FIFO (*First in first out*) A operação que chega primeiro ao centro de trabalho é processada primeiro.

EDD (*Earliest Due Date*) As operações são processadas por ordem crescente das respectivas datas devidas de entrega das ordens de fabrico a que pertencem. Frequentemente "*earliest*" é omitido.

LWKR (*Least Work Remaining*) A prioridade de processamento é dada à ordem de fabrico com o menor valor da soma das durações das operações por realizar.

MWKR (*Most Work Remaining*) A prioridade de processamento é dada à operação cuja ordem de fabrico tem o maior valor da soma das durações das operações por realizar.

RANDOM (*Random Selection*) Esta regra selecciona a próxima operação a ser processada, aleatoriamente. Esta regra não deve ser utilizada na prática constituindo apenas um termo de comparação com outras regras.

MOPNR (*Most Operations Remaining*) Selecciona a operação cuja ordem de fabrico tem o maior número de operações por realizar.

CR (*Critical Ratio*) É uma regra dinâmica que fornece um número indicador de prioridade que exprime a relação tempo restante por trabalho restante (Monks, [1987]). Esta regra pode ser constantemente actualizada. Se CR for menor que 1.0 a ordem está atrasada, se CR = 1.0 está em dia, se CR for maior que 1.0 a ordem de fabrico tem alguma folga.

$$\text{CR} = \frac{\text{data de entrega} - \text{data actual}}{\text{trabalho restante}}$$

1/C Selecciona a ordem de fabrico que tem a maior penalidade por atraso.

VALUE Selecciona-se a ordem de fabrico com o maior valor.

COVERT (*Cost Over Time*) Processa-se em primeiro lugar a ordem de fabrico com maior cociente entre o custo associado ao valor esperado do atraso e o tempo de processamento da operação.

COST/TIME Primeiro selecciona uma ordem de fabrico crítica usando SPT, de seguida selecciona uma ordem que está atrasada, mas não crítica, com a maior taxa de custo de atraso por tempo de operação; de seguida selecciona por SPT (Esta regra é similar a COVERT).

WINK (*Work In Next Queue*) É processada em primeiro lugar a ordem de fabrico que utilizar na operação seguinte a máquina com menor trabalho.

NINQ (*Number of Jobs In Near Queue*) Selecciona a ordem que vai para a máquina com a mais pequena fila de trabalho.

SHOPNH Selecciona a ordem por SPT, avaliando se está para chegar alguma ordem de fabrico com menor tempo de processamento. Se sim a máquina aguarda a chegada dessa ordem para a processar.

Observe-se que as regras WINK e NINQ são regras globais, porque têm em consideração a informação relativa a todas as máquinas. Estas duas regras não incorporam no entanto a data de entrega das ordens de fabrico. Blackstone jnr *et al.* (1982), afirmam que estas duas regras globais têm um desempenho inferior ao das regras que utilizam datas de entrega.

A selecção de regras de prioridade envolve situações de conflito entre vários critérios de desempenho: serviço ao cliente, lucros, urgência das encomendas, utilização das máquinas, etc. Não existe nenhuma regra melhor do que outra para todos os critérios de desempenho, Tersine (1987).

.3.8 Metaheurísticas

Nas secção seguintes apresentam-se as seguintes abordagens de metaheurísticas: *tabu search, simulated annealing* e algoritmos genéticos.

..3.8.1 *Tabu search*

Os algoritmos baseados em *tabu search* têm sido aplicados a problemas de planeamento de operações em ambiente *job shop* por diversos autores. Salientam-se nesta área os trabalhos de Dell'Amico (1993), Taillard (1994), Barnes e Chambers (1995), Lourenço e Zwijnenburg (1996) e Nowicki e Smutnicki (1996).

1.3.8.2 *Simulated annealing*

Os algoritmos baseados no *simulated annealing* têm sido aplicados a problemas de planeamento de operações em ambiente *job shop* por diversos autores. Salientam-se nesta área os trabalhos de Laarhoven *et al.* (1992), Lourenço (1995) e Kolonko (1999).

4.3.8.3 Algoritmos genéticos

Os algoritmos genéticos têm sido aplicados a problemas de planeamento de operações em ambiente *job shop* por diversos autores.

Kobayashi, Ono e Yamamura (1996) apresentaram alguns resultados interessantes utilizando um operador de cruzamento de troca de subsequências. Usam

uma codificação literal para representar a sequência em cada máquina. Só permitem cruzamentos de ambos os "pais", têm uma subsequência que contêm as mesmas operações para uma ou mais máquinas. Procedem então à alteração das sequências resultantes em activas usando o método de geração de planos activos de Giffler e Thompson.

Croce, Tadei e Volta (1995), descrevem um algoritmo genético que utiliza uma codificação directa de maneira a determinar a lista de preferências para a programação de operações para cada máquina. Para evitar problemas de inviabilidade, os cruzamentos só podem ocorrer entre diferentes máquinas. Uma limitação do seu mecanismo de escalonamento é que apenas constrói planos do tipo não-atrasados. Em problemas *Job Shop* o melhor plano não-atrasado pode ter um valor da função objectivo que é muito superior ao valor óptimo. Para aliviar esta dificuldade os autores introduziram um mecanismo de *"lookahead"* para inserir atrasos no plano.

Tanto Fox e McMahon (1991) como Nakano (1991) utilizam um algoritmo genético binário onde o cromossoma contém um *bit* para cada par de ordens de fabrico possível. A precedência das ordens de fabrico é determinada pela presença de um 0 ou 1 para o gene. Esta representação também resulta em cromossomas que se traduzem em planos inviáveis que seguem o uso dos operadores de cruzamento *single-point, multi-point* e uniforme. Nakano utiliza os mecanismos de "harmonização" e "pressão" para assegurar que todos os cromossomas são traduzidos em sequências viáveis. Fox e McMahon desenvolveram dois novos operadores a que chamam "união" e "intersecção" de forma a partilharem a informação dos genes pelos cromossomas.

Fang, Ross e Crone (1993) aplicaram ao problema JSP uma codificação ordenada. Dado um cromossoma de comprimento L, os valores possíveis para cada gene, g, são os inteiros compreendidos entre 1 e $L+1-g$. Esta codificação tem a vantagem de o cruzamento poder ser aplicado a cada dois cromossomas e a geração resultante será viável. Apesar disto, este método pode ser pouco eficiente porque a interpretação dos genes localizados no fim do cromossoma é altamente dependente dos valores dos genes localizados no início do cromossoma. Uma mudança no valor de um gene no início de um cromossoma pode ter um grande efeito no plano resultante.

Dorndorf e Pesch (1995) apresentaram duas aplicações de algoritmos genéticos. A primeira aplicação constrói planos activos utilizando o algoritmo de Giffler e Thompson (1960). Em cada fase do algoritmo é gerado um conjunto de operações em conflito. Os conflitos gerados na escolha das operações a programar são decididos com recurso a regras de prioridade que por sua vez são seleccionadas pelo algoritmo genético. A segunda aplicação utiliza o algoritmo genético para a

determinação da ordem das máquinas críticas. Esta informação é incorporada na resolução do problema através do algoritmo *shifting bottleneck* descrito por Adams, Balas e Zawack (1988).

Norman e Bean (1997) utilizaram um esquema de representação baseado em chaves aleatórias. O algoritmo resume-se a considerar valores aleatórios que estabelecem as prioridades das operações.

Oliveira (2000) apresenta um algoritmo genético para o problema JSP que utiliza o algoritmo de Giffler e Thompson (1960) para gerar planos activos. A escolha das operações a programar é realizada a partir das prioridades geradas pelo algoritmo genético. O autor integra adicionalmente um procedimento de pesquisa local.

4 Outras abordagens

A dificuldade na resolução do problema JSP, tem levado alguns autores a procurar encontrar outras abordagens para a resolução deste problema. De salientar, por exemplo, Zhou *et al.* (2001), Wang e Zheng (2001), Binato *et al.* (2002) e Aiex *et al.* (2003).

Zhou *et al.* (2001) propõem uma heurística híbrida utilizando um algoritmo genético e regras de prioridade tais como SPT e MWKR. As regras de prioridade são integradas no processo de evolução genética. Adicionalmente, os autores propõem a utilização de uma técnica de melhoria local.

Wang e Zheng (2001) propõem também uma abordagem híbrida, utilizando um algoritmo genético e *simulated annealing*. Os autores utilizam o algoritmo de *simulated annealing* para controlar o processo de evolução genética.

Binato *et al.* (2002) propõem uma metaheurística de natureza combinatorial GRASP (*Greedy Randomized Adaptive Search Procedure*) com melhoria local, e a utilização da abordagem QAP (*Quadratic Assignement Problem*) proposta por Fleurent e Glover (1999). A utilização da abordagem QAP pretende melhorar o processo iterativo do GRASP, incorporando um esquema de memória no processo de evolução.

03

ALGORITMOS GENÉTICOS

Introdução

Dado os algoritmos desenvolvidos para os problemas RCPSP, RCMPSP e JSP nos capítulos 5, 6 e 7 serem todos baseados em algoritmos genéticos far-se-á neste capítulo uma breve revisão sobre este tipo de algoritmos.

Os algoritmos genéticos referem-se a uma classe de procedimentos de procura adaptativa, inspirada nos princípios evolutivos das populações genéticas da natureza e foram concebidos por Holland (*Michigan University*) nos anos 60. A sua denominação tem origem na analogia entre a representação de uma solução e o conceito da estrutura genética de um cromossoma.

Tanto no contexto da genética natural como no contexto dos algoritmos genéticos, existem dois conceitos fundamentais:

Evolução através da geração de novas populações;
Adaptação processo através do qual uma população é progressivamente modificada por forma a obter um melhor desempenho no seu ambiente.

Durante o curso da "evolução" as populações naturais evoluem de acordo com os princípios da selecção natural e da adaptação. Ou seja, indivíduos que se adaptam melhor ao ambiente, possuem maiores hipóteses de sobreviver e de se reproduzirem, enquanto que os indivíduos menos adaptados tendem a ser eliminados. Isto significa que os genes dos indivíduos altamente aptos influenciarão um número crescente de indivíduos em cada geração e a sua espécie evoluirá de forma a tornar-se cada vez mais adaptada ao seu ambiente.

Um algoritmo genético simula o processo de "adaptação" considerando uma população inicial de indivíduos e aplicando operadores genéticos artificiais em cada geração. No caso da resolução de problemas de optimização, cada indivíduo da população é codificado como um cromossoma, o qual representa uma solução possível para o problema. A adaptação dos indivíduos é avaliada através de uma função de mérito, que determina a aptidão de cada indivíduo. Aos indivíduos altamente aptos (soluções com melhor mérito) são dadas maiores oportunidades de se reproduzirem trocando partes de informação genética, num procedimento de acasalamento denominado "cruzamento"; o operador de "mutação" é utilizado para alterar alguns genes nos cromossomas e assim aumentar a diversidade na população. A descendência ou nova população, pode substituir toda a população actual ou substituir apenas os indivíduos de menor aptidão. Este ciclo de avalia-

ção, selecção e transformação das populações é repetido sucessivamente até que uma solução satisfatória seja encontrada.

3.2 Revisão da literatura

O desenvolvimento inicial desta técnica ocorreu nas décadas de 60 e 70, e a primeira descrição rigorosa foi feita por Holland no livro *Adaptation in Natural and Artificial Systems* (1975), no qual o autor também demonstrou o sucesso da aplicação a uma variedade de problemas, tais como reconhecimento de padrões, sistemas de classificação, configuração de redes de comunicação, sistemas de controlo industrial e optimização combinatória em geral. Ainda em termos de literatura básica no ano de 1975, De Jong completou a sua tese de doutoramento *An Analysis of the Behavior of a Class of Genetic Adaptative Systems*, combinando a teoria dos *Schemata* de Holland e as suas experiências computacionais; Liepins e Hilliard (1989), apresentam uma boa revisão dos maiores desenvolvimentos de aplicação; Goldberg (1989), Beasley *et al.* (1993) e Reeves (1993) fazem avaliações e demonstram o conhecimento básico dos algoritmos genéticos; finalmente, uma abordagem mostrando as dificuldades de aplicação a algumas áreas, pode ser encontrada em Reeves (1991).

As primeiras aplicações para problemas de optimização matemática bem definidos, foram apresentados por Hollstien (1971), seguido por De Jong (1975), com uma análise do comportamento de uma classe de sistemas adaptativos genéticos.

No campo da Investigação Operacional relacionado com a modelação através de redes, Goldberg e Lingle Jr. (1985), Grefenstette *et al.* (1985) e Whitley *et al.* (1991) apresentaram importantes trabalhos ligados ao "*Travelling Salesman Problem*" e Kapsalis *et al.* (1993) desenvolveram um algoritmo genético para o Problema da Árvore de Steiner. Comparações interessantes sobre as performances de algoritmos genéticos, *Simulated Annealing* e um algoritmo de vizinhança simples, podem ser encontradas em Reeves (1995).

Os recentes avanços teóricos têm permitido a aplicação dos algoritmos genéticos a problemas industriais. Smith (1983) e Schaffer e Grefenstette (1985) utilizaram estes algoritmos em "*machine learning*", Goldberg (1984) utilizou no controlo de sistemas de operações em oleodutos e Fourman (1985) resolveu um problema de *Layout* VLSI.

Na área do planeamento da produção, destacam-se os trabalhos de Davis (1985) e Biegal e Davern (1990) nos problemas *Job Shop*; Cleveland e Smith (1989), Gauthier (1993) e Neppalli *et al.* (1996) no problema de programação *Flow Shop*; Venugopal e Narendran (1992) propõem um modelo matemático multi-objectivo para o problema da formação de células de fabrico; Levitin e Rubinovitz (1993)

sugerem uma aplicação para resolver os casos *Linear Assignment Problem* (LAP) e *Ciclic Assignment Problem* (CAP) do *Quadratic Assignment Problem*; e López Vaca (1995) apresenta um algoritmo para a programação de projectos com minimização do *makespan* e nivelamento de recursos.

Na investigação sobre o mais importante dos operadores genéticos, o cruzamento (*Crossover*), tem sido particularmente interessante o estudo das redundâncias e das informações perdidas, que podem resultar das permutações feitas pelo operador utilizado. Alguns esquemas foram desenvolvidos para evitar as soluções não admissíveis, tais como o operador OX (*Order Crossover*) (Davis (1991)), o operador PMX (*Partially Mapped Crossover*) (Goldberg (1989)) e o operador CX (*Cycle Crossover*) (Goldberg (1989)). Seguindo outra linha orientadora, Beasley e Chu (1996) defendem uma taxa variável de mutação e propõem um novo operador de cruzamento (fusão), baseado na aptidão (*fitness*).

Finalmente, é interessante observar que a literatura técnica dos algoritmos genéticos não demonstra de forma conclusiva o desempenho desta técnica em relação a outros métodos aproximados. As análises experimentais e teóricas tem sido geralmente dirigidas para a solução de um problema específico.

Quando utilizar um algoritmo genético

Não existe consenso sobre o campo de aplicabilidade dos algoritmos genéticos. No entanto, existe concordância sobre a aplicação de algoritmos genéticos nos seguintes aspectos:

O espaço a ser pesquisado é de grande dimensão se o espaço a ser pesquisado for de grande dimensão, não será viável pesquisá-lo exaustivamente até que seja encontrada uma solução óptima global;

A função a ser optimizada não é unimodal dada a possibilidade de neste caso existirem múltiplos óptimos locais, os algoritmos genéticos permitem evitar com sucesso a convergência para óptimos locais;

O problema não requer solução óptima quando o utilizador tem por objectivo apenas uma solução de boa qualidade (próxima do óptimo) e com reduzido tempo computacional.

4 Vantagens dos algoritmos genéticos

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

Não linearidade enquanto as técnicas de optimização convencionais se baseiam em pressupostos irrealistas de linearidade, convexidade e diferencialidade, entre outros, os algoritmos genéticos não utilizam nenhum desses pressupostos;

Robustez como consequência da independência de domínio e da não linearidade, os algoritmos genéticos podem resolver uma diversidade de tipos de problemas, bem como podem trabalhar com funções não-lineares;

Facilidade de modificação as modificações de um algoritmo genético para modelar variações do problema original são muito fáceis de ser implementadas, ao contrário do que acontece em muitos outros métodos;

Natureza paralela um algoritmo genético possui natureza tal que facilita a efectivação de implementações com processamento paralelo. Por exemplo, assumindo a substituição em bloco, a aptidão de cada cromossoma poderia ser calculada em paralelo, reduzindo o tempo de processamento de cada geração.

3.5 A robustez dos algoritmos genéticos

Além dos argumentos apresentados na secção anterior, existem outros argumentos relativos à robustez dos algoritmos genéticos em relação aos métodos tradicionais de optimização:

- Os algoritmos genéticos são probabilísticos;
- A propriedade do paralelismo intrínseco, através da qual os algoritmos genéticos permitem a simultaneidade da avaliação das soluções;
- A diversidade genética, isto é, a cada passo do processo de procura, um conjunto de soluções candidatas é considerado e vários elementos são simultaneamente envolvidos na geração de novas soluções candidatas;
- Os algoritmos genéticos avaliam a aptidão dos candidatos; ou seja, o ajuste não é necessariamente derivado de uma função objectivo a ser optimizada;
- O uso da codificação de um conjunto de variáveis, em lugar das próprias variáveis;
- Um algoritmo genético constitui um método de procura aleatória e portanto funcionará melhor do que heurísticas comuns quando se dispõe de pouca informação sobre o espaço de procura.

3.6 Terminologia dos algoritmos genéticos

A terminologia utilizada pelos algoritmos genéticos consiste basicamente de analogias extraídas da Biologia. Os termos mais utilizados são, segundo Mitchel (1996):

Aptidão (*Fitness*) probabilidade que um organismo possui de se reproduzir (viabilidade), ou uma função do número de descendentes (fertilidade) que este organismo possui;

Cromossoma (*Chromosome*) uma estrutura de solução candidata para o problema;

População (*Population*) conjunto dos cromossomas que compõe cada geração;

Gene (*Gene*) divisão conceptual ou bloco funcional de um cromossoma, capaz de codificar uma característica;

Posição (*Locus*) posição em que um gene se localiza no cromossoma;

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

Cruzamento (*Crossover*) troca de partes entre dois cromossomas;

Mutação (*Mutation*) mudança ou troca de um ou mais alelos de um cromossoma.

7 Representação de uma solução

De acordo com a biologia, cromossomas são extensas cadeias de compostos químicos que contém a descrição da composição genética dos seres vivos. Do ponto de vista dos algoritmos genéticos, um cromossoma codifica uma solução do problema, ou seja, um ponto no espaço das soluções.

Originalmente e em muitas aplicações um cromossoma consiste numa sequência de codificação binária, ver **fig. 3.1**, sendo que a presença de "1" ou "0" numa determinada posição significa a existência ou não de uma determinada característica. Esta codificação aparentemente simples não é única.

fig. 3.1 Exemplo de estrutura de um cromossoma em codificação binária.

011100110

As soluções candidatas podem também ser implementadas em codificações que usam alfabetos não binários, com o objectivo de melhorar a adaptação ao tipo de problema abordado. No entanto, a investigação teórica existente para codificações não binárias é bastante reduzida quando comparada com a codificação original. Estudos comparativos entre as várias codificações foram realizados por Janikow e Michalewicz (1991) e O'Reilly e Oppacher (1995).

3.7.1 Tipos de representação

Uma característica importante que distingue os algoritmos genéticos dos algoritmos convencionais é o facto de o algoritmo não trabalhar directamente com as soluções do problema, mas sim com uma representação destas, os cromossomas.

Ao longo dos recentes anos de investigação sobre algoritmos genéticos, desenvolveram-se uma razoável variedade de representações que podem ser classificadas em dois tipos: a representação indirecta e a representação directa.

A representação indirecta é a representação que é utilizada com mais frequência. Neste caso o algoritmo genético trabalha sobre uma população de cromossomas à qual é necessário aplicar uma função descodificadora para obter as soluções correspondentes.

Na representação indirecta podem-se considerar, por um lado, os modelos independentes do domínio do problema e, por outro lado, os modelos específicos do problema, Bruns (1993). A principal característica do modelo independente do domínio do problema é o facto de se poderem usar os operadores genéticos convencionais. Os modelos específicos do problema utilizam os operadores genéticos especificamente desenvolvidos para o problema os quais incorporam informação específica sobre este.

Na representação directa o cromossoma define a solução, dispensando quaisquer procedimento de descodificação de soluções.

3.8 Schemata

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

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

9 Avaliação das soluções candidatas

A aptidão (*fitness*) refere-se ao grau com que uma solução candidata contribui para a convergência do algoritmo na procura da melhor solução. Para medir esta grandeza utiliza-se uma função de aptidão que tem por objectivo estabelecer uma medida de qualidade para o cromossoma. Como exemplo, no caso em que o cromossoma utiliza codificação binária, o cálculo da aptidão (*fitness*) poderia consistir em fazer a descodificação do cromossoma para um número real e então avaliar a função objectivo para aquele ponto.

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

Convergência prematura A dominância dos cromossomas com alto valor de aptidão nas populações iniciais, pode fazer com que o algoritmo venha a convergir muito rapidamente para um ponto de máximo local no espaço de soluções. Deste facto, surge a necessidade de modificar a função objectivo em prol da contribuição dos indivíduos menos aptos, através de vários mecanismos, de entre os quais se pode citar o procedimento de escala, o procedimento de *rank* ou o procedimento de torneio (os dois últimos serão mostrados no item referente à selecção). A ideia básica do procedimento de escala é limitar a competição entre os cromossomas nas iterações iniciais e estimulá-la progressivamente, através de uma mudança (linear em geral) de escala na aptidão. No caso da mudança linear de escala, que se descreve, os valores a e b são obtidos a partir das condições:

$$f'_{médio} = Y_{médio} \quad \text{e} \quad f_{máx} = K * Y_{médio}$$

o parâmetro K é o número esperado de descendentes do melhor cromossoma, usado para assegurar que o membro mais ajustado da população será escolhido K vezes na média. Para pequenas populações ($n = 50$ até 100), $Y = 1{,}2$ a 2 representa um intervalo de valores já utilizado com sucesso. A relação linear entre f e Y pode ser representada da seguinte forma: $f = a * Y + b$

onde

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

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

3.10 Operadores genéticos

Os operadores genéticos são mecanismos que permitem causar variação nos esquemas existentes. Analisando sob o ponto de vista matemático, os operadores genéticos são operadores cuja função é criar novos pontos de procura no espaço de soluções, com base nos elementos da população actual. Existem vários operadores para a manipulação dos cromossomas, sendo os mais comuns a Seleção (*Selection*), o Cruzamento (*Crossover*), a Mutação (*Mutation*) e a Inversão (*Inversion*).

3.10.1 Selecção

O operador de selecção dos algoritmos genéticos é uma analogia com a selecção biológica natural de Darwin. Realizar uma selecção significa escolher de entre os indivíduos pertencentes à população, aqueles que servirão para criar descendentes para a próxima geração e quantos descendentes cada um deverá gerar. Uma selecção pode ser denominada muito fraca, se resultar numa evolução muito baixa e muito forte se resultar em populações de indivíduos mais fortes (o termo forte aqui utilizado, refere-se aos indivíduos mais adaptados ao meio ambiente; esta adaptação é que orienta o processo evolutivo). Isto significa que os indivíduos mais adaptados vivem mais e geram mais descendentes, os quais herdam as suas qualidades. Fica portanto evidente, que o mecanismo da selecção funciona como um filtro dos indivíduos de uma população, para aumentar a perspectiva de gerar melhores indivíduos (segundo determinado critério).

3.10.1.1 Mecanismos de selecção

Um algoritmo genético funciona com base na manutenção de uma população de cromossomas, que são os progenitores potenciais. A escolha dos progenitores pode ser feita de inúmeras formas e de entre os mecanismos mais importantes, figuram a selecção através da proporcionalidade do *fitness*, a selecção Boltzmann, a mudança de escala sigma e a selecção *rank*.

Selecção através da proporcionalidade do *fitness* (*Fitness-Proportionate Selection*) Esta denominação atribui-se a todo processo para o qual o valor esperado do indivíduo é resultante da razão entre o seu *fitness* e o *fitness* médio da população. O método mais comum para realizar este operador é a selecção pelo giro da roleta (*Roulette Wheel Selection*). Neste tipo de selecção, análogo ao giro de uma roleta, denominada por Brindle (1981) de amostragem probabilística com substituição (*Stochastic Sampling with Replacement*, Goldberg (1989)), cada cromossoma possui uma probabilidade de ser escolhido proporcional à sua aptidão. Portanto, cada indivíduo possui a probabilidade:

$$P_{select_i} = \frac{f_i}{\sum f_i}$$

de ser escolhido, onde f_i é a sua aptidão, i.e., cada indivíduo da população possui um número de fatias da roleta proporcional à sua adaptação.

Segundo Mitchel (1996), quando se utiliza um processo de selecção por proporcionalidade do *fitness*, a convergência tende a ser prematura, visto que é dada muita ênfase desde as primeiras iterações, na exploração de indivíduos de *fitness* mais altos, dispensando outras regiões do espaço de procura. Posteriormente, quando existe muita semelhança entre os indivíduos da população, não existem diferenças de *fitness* que sejam capazes de continuar a evolução. Ao poder que faz com que os indivíduos com maior *fitness* tenham mais descendentes, denomina-se pressão de selecção no vocabulário dos algoritmos genéticos.

A conclusão é que a taxa de evolução depende da variância do *fitness* da população. Neste sentido, diversos investigadores têm proposto outros tipos de selecção, que transformem os valores originais dos *fitnesses* em valores esperados, para evitar a convergência prematura. Assim surgiram entre outros, operadores como a selecção Boltzmann, a mudança de escala sigma e a selecção *rank*.

Selecção Boltzmann (*Boltzmann Selection*) O método de selecção Boltzmann, Goldberg (1990); extraído do *simulated annealing*, estabelece uma pressão de selecção diferente em cada instante da procura de soluções. Inicialmente admite a reprodução de indivíduos com *fitness* baixo, permitindo assim manter a diversidade da população e evitar convergências prematuras, e posteriormente aumenta a pressão para dar ênfase aos indivíduos com *fitness* alto. Segundo a analogia estabelecida, a contínua variação da temperatura controla a taxa de selecção: quando a temperatura é alta, a pressão de selecção é baixa (todos os indivíduos possuem a mesma probabilidade de reprodução); a temperatura é gradualmente reduzida, incrementando a pressão de selecção, forçando que o

algoritmo se limite à melhor parte do espaço de pesquisa, mantendo um grau apropriado de diversidade. Na expressão abaixo, verifica-se que quando a temperatura T decresce, a diferença no valor esperado $VE(i,t)$ entre um *fitness* $f(i)$ baixo e outro alto é aumentada.

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

onde

VE = valor esperado do indivíduo i no tempo t;
T = temperatura;
$(e^{f(i)/T})_t$ = média sobre a população i no tempo t.

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

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

onde

$VE(i,t)$ = valor esperado do indivíduo i no tempo t;
$f(i)$ = *fitness* do indivíduo i;
$f'(t)$ = *fitness* médio da população no tempo t;
$\sigma(t)$ = desvio padrão do *fitness* da população no tempo t.

Selecção rank (*Rank Selection*) O principal objectivo deste operador é evitar a convergência muito rápida. Segundo a versão linear proposta por Ficici *et al.* (2000), os indivíduos da população são ordenados crescentemente de acordo com o seu mérito. De seguida um novo mérito é atribuído, $f(i) = i$, a cada elemento que varia de 1 a N. Após a atribuição deste novo mérito, aplica-se o giro da roleta. Desta forma o valor esperado depende desta nova ordem e não do valor absoluto do mérito, sendo que:

$$\sum_i VE(i,t) = N$$

Após este passo, o valor esperado de cada indivíduo i da população no tempo t é definido por:

$$VE(i,t) = \frac{f_i}{\sum f_i} x N$$

onde

$VE(i,t)$ = valor esperado do indivíduo i no tempo t;
$f(i)$ = mérito do indivíduo de ordem i.

3.10.1.2 O processo de substituição dos indivíduos

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

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

Substituição em estado estacionário o tipo de reprodução dita em estado estacionário, Davis (1989), prevê a substituição gradual dos cromossomas de uma população actual. Em cada iteração do algoritmo, um ou mais cromossomas são escolhidos para dar lugar aos descendentes. Este mecanismo foi elaborado a partir da necessidade de corrigir algumas distorções de outros operadores, como a destruição ou alteração de muitas características interessantes de uma população, quando os seus elementos são submetidos aos operadores de cruzamento e mutação; bem como foi elaborado para evitar a exclusão de muitos dos bons indivíduos que não se reproduzem, quando submetidos à Reprodução Tradicional. Quando o processo só aceita a substituição se o candidato descendente for diferente dos cromossomas da população actual, a técnica recebe a denominação especial de Reprodução em Estado Estacionário Sem Duplicação.

3.10.2 Operador de cruzamento

Muitos autores consideram que este operador é a principal particularidade dos algoritmos genéticos em relação aos outros métodos aproximados. Após dois cro-

mossomas serem seleccionados, o cruzamento pode ser realizado se a sua taxa ou probabilidade pré-definida de realização for atingida.

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

O cruzamento com um ponto de corte é feito através da escolha aleatória de um só ponto de cruzamento. Cada par de cromossomas escolhidos como progenitores gera dois descendentes através da permuta das suas partes finais, depois do ponto de cruzamento. A ideia é recombinar esquemas em diferentes cromossomas. Os três principais problemas deste tipo de operador são: (1) não poder combinar todos os esquemas possíveis; (2) esquemas de grande dimensão são sujeitos à destruição; ou seja, a criação ou destruição de um esquema depende fortemente da localização do bit no cromossoma, Eshelman e Schaffer (1991) e (3) os descendentes de dois progenitores, contêm sempre os pontos finais dos cromossomas, caracterizando uma tendência de sobrevivência das partes finais.

Para exemplificar o cruzamento de um único ponto, os seguintes cromossomas denominados C1 e C2, a partir de uma posição gerada aleatoriamente igual a 4, darão origem aos descendentes D1 e D2, conforme a ilustração a seguir:

C1	0	1	1	1	0	0	1	1	0
C2	1	0	1	0	1	1	1	1	0

A nova configuração seria:

D1	0	1	1	0	1	1	1	1	0
D2	1	0	1	1	0	0	1	1	0

O cruzamento de dois pontos resultou da necessidade de reduzir os problemas do anterior. Neste tipo de cruzamento são escolhidas duas posições aleatoriamente e os segmentos entre elas são trocados. Desta forma, fica menos provável a destruição de esquemas de grande dimensão, bem como os segmentos trocados nem

sempre contém os pontos extremos dos cromossomas. Utilizando os cromossomas C1 e C2, e utilizando duas posições aleatórias 3 e 6, obtêm-se as configurações D1 e D2 ilustradas a seguir:

C1	0	1	1	1	0	0	1	0	1
C2	1	0	1	0	1	1	1	1	0

A nova configuração seria:

D1	0	1	1	0	1	1	1	0	0
D2	1	0	1	1	0	0	1	1	0

Em problemas como o *Traveling Salesman Problem* e o *Project Scheduling Problem*, nos quais a modelação é feita através de redes, a codificação do cromossoma é em geral realizada através de números inteiros, que representam as cidades e as actividades, respectivamente. O resultado é que os dois tipos de cruzamento anteriormente mostrados, não são adequados devido às repetições que resultam da operação. Vários operadores foram criados para contornar esta situação, e neste trabalho apresentam-se quatro tipos: o operador OX (*Order Crossover*), o operador PMX (*Partially Mapped Crossover*), o operador CX (*Cycle Crossover*) e o operador LOX (*Linear Order Crossover*), ilustrados a seguir em exemplos simples.

Operador PMX Considerando-se os seguintes cromossomas C1 e C2, inicialmente dois pontos de corte são escolhidos ao acaso e os genes presentes entre os pontos de corte [4-7-5] e [6-2-8] são trocados de forma que ambos recebem partes de sequência de informações genéticas novas.

C1	3	2	8	4	7	5	9	1	6
C2	4	9	5	6	2	8	3	7	1

A nova configuração seria:

D1	3	2	8	6	2	8	9	1	6
D2	4	9	5	4	7	5	3	7	1

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

D1	3	5	7	6	2	8	9	1	4
D2	8	9	2	4	7	5	3	6	1

Operador ox O operador ox inicia de forma semelhante ao pmx, com a escolha aleatória de dois pontos de corte e transferência integral dos genes internos entre os progenitores. Utilizando os mesmos cromossomas C1 e C2 do exemplo anterior, o procedimento consiste em transferir a sequência anterior ao primeiro ponto de corte, para o final do cromossoma.

C1	3	5	7	6	2	8	9	1	4	3	5	7
C2	8	9	2	4	7	5	3	6	1	8	9	2

A seguir, troca-se a sequência anterior ao primeiro ponto de corte de um progenitor, pela sequência entre os pontos de corte do outro progenitor. No exemplo, a sequência [3-5-7] de C1, foi trocada pela sequência [4-7-5] de C2, e a sequência [8-9-2] de C2, foi trocada pela sequência [6-2-8] de C1. Marca-se em negrito os genes repetidos após o segundo ponto de corte, obtendo-se os cromossomas D1 e D2:

D1	4	7	5	6	2	8	9	1	**4**	3	**5**	**7**
D2	6	2	8	4	7	5	3	**6**	1	**8**	9	**2**

Finalmente, elimina-se os genes repetidos marcados:

D1	4	7	5	6	2	8	9	1	3
D2	6	2	8	4	7	5	3	1	9

Operador cx O operador cx possui um procedimento muito diferente dos operadores pmx e ox, visto que não utiliza pontos de corte. O operador executa a recombinação de modo que cada gene das descendências vem das posições correspondentes de qualquer um dos dois progenitores. Usando novamente os cromossomas C1 e C2, começa-se da esquerda e escolhe-se um gene do primeiro progenitor. O descendente D1 leva o valor do primeiro gene de C1:

D1	3	?	?	?	?	?	?	?	?

O gene de mesma posição em C2 possui valor 4. Procurando-se este valor em C1, verifica-se que ele se encontra por coincidência no quarto gene, e este valor deve ser levado para a quarta posição de D1:

D1 3 ? ? 4 ? ? ? ? ?

A seguir, verifica-se que o gene de mesma posição em C2 possui valor 6 e que este valor encontra-se na nona posição de C1. Assim, este valor é transportado para a nona posição de D1:

D1 3 ? ? 4 ? ? ? ? 6

O gene que ocupa a mesma posição em C2, possui valor 1, o qual se encontra na oitava posição em C1. Transporta-se este valor para D1:

D1 3 ? ? 4 ? ? ? 1 6

Com a continuação do processo, coloca-se o valor 7 no quinto gene de D1; o valor 2 no segundo gene; o valor 9 no sétimo gene. Mas ao tentar colocar o valor 3 no primeiro gene, observa-se que esta operação já foi realizada.

D1 3 ? ? 4 ? ? ? 1 6

Portanto o ciclo está completo, e o passo seguinte é completar os genes de D1, com os respectivos elementos de C2:

D1 3 2 ? 4 7 ? 9 1 6

O segundo filho é obtido através do mesmo processo, começando por C2, e resulta em:

D1 3 2 5 4 7 8 9 1 6

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

Operador LOX O cruzamento de ordem linear LOX, uma variante do operador OX, foi elaborado por Faulkenauer e Bouffoix (1991), para satisfazer a todos os

requisitos que o JSP exige. Para exemplificá-lo, consideraram-se dois pontos aleatórios de cruzamento nos cromossomas C1 e C2:

C1	1	2	3	4	5	6	7	8	9
C2	7	6	9	4	3	2	1	5	8

Os valores de C2 que são comuns com os valores entre os pontos de cruzamento de C1 [4, 5 e 6], são substituídos por asteriscos:

D2	7	*	9	*	3	2	1	*	8

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

D2	7	9	3	*	*	*	2	1	8

Transferir o segmento de C1, localizado entre os pontos de corte, para a mesma região de D2. Para gerar o descendente D1 é realizada a mesma sequência de operações invertendo os papéis.

D1	1	5	6	4	3	2	7	8	9

3.10.3 Operador de mutação

A mutação é considerada um mecanismo secundário nas operações dos algoritmos genéticos, consistindo nas mudanças aleatórias de um ou mais genes de um cromossoma, e consequentemente produzindo indivíduos com algumas propriedades de cromossomas completamente diferentes da maioria de uma população. A sua função é criar uma política de prevenção que diminua as probabilidades de as soluções convergirem para um óptimo local. A taxa de mutação é normalmente fixada num nível muito baixo, tal como nas populações naturais.

A mutação simples – *Simple Mutation* – Goldberg (1989) consiste na escolha aleatória de uma posição no cromossoma de codificação binária seleccionado, e de um sorteio entre os alelos "0" e "1", como exemplificado a seguir, utilizando a terceira posição no cromossoma C1.

C1	1	1	0	1	0	0	1	0	1

A nova configuração poderia ser:

D1	1	1	1	1	0	0	1	0	1

Segundo Davis (1991), podem ainda considerar-se três tipos de mutação, para problemas em que a ordem de elementos dos cromossomas é uma grandeza importante: *Position-Based Mutation, Order-Based Mutation e a Scramble Mutation.*

Position-Based Mutation Duas posições são seleccionadas aleatoriamente e o conteúdo da segunda posição é colocado antes da primeira:

C1	5	1	3	8	5	4	2	6	7

A nova configuração poderia ser:

D1	5	1	6	3	8	5	4	2	7

Order-Based Mutation Duas posições são seleccionadas aleatoriamente e os seus conteúdos são trocados:

C1	5	1	3	8	5	4	2	6	7

A nova configuração é:

D1	5	1	6	8	5	4	2	3	7

Scramble Mutation Supondo que a vizinhança das posições é importante, escolhe-se uma sublista aleatoriamente e permuta-se a ordem dos conteúdos:

C1	5	1	3	8	5	4	2	6	7

A nova configuração poderia ser:

D1	5	1	8	4	3	5	2	3	7

3.10.4 Operador de inversão

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

Este facto deu origem à criação dos operadores de reordenação, particularmente o operador de inversão. Na inversão, uma secção do cromossoma é retirada e reinserida na ordem inversa (ver o exemplo a seguir, onde o segmento sombreado de C1 é invertido). Segundo Reeves (1993), quando utilizado fora desta situação, o operador de inversão deve ter a mesma finalidade da mutação e permitir a exploração de regiões do espaço de procura que o cruzamento não alcançará rapidamente. Em geral não se pode perder de vista que o operador de inversão não possui a mesma força de recombinação que o operador de cruzamento.

C1	1	1	0	1	0	0	1	0	1

A nova configuração seria:

D1	1	1	0	0	1	0	1	0	1

3.11 Estratégias de reprodução

A tese de De Jong (1975) e o trabalho de Goldberg (1989) constituem as principais referências na literatura dos algoritmos genéticos sobre estratégias de reprodução. O objectivo do trabalho de De Jong (1975) foi o de estudar as variações do algoritmo genético a partir de uma versão por ele denominada de estratégia reprodutiva R1 (que Goldberg em 1989, denomina *Simple Genetic Algorithm*), com as seguintes características:

- Codificação binária do cromossoma;
- Selecção pelo giro da roleta;
- Cruzamento simples (de um único ponto e acasalamento aleatório);
- Mutação simples;
- Substituição total de todos os cromossomas de uma população pelos seus descendentes, a fim de compor a nova população.

De Jong estudou as variações da estratégia R1 e provou que a estratégia não era uma única, mas uma família de estratégias que dependem de quatro parâmetros:

tamanho da população, probabilidade de cruzamento, probabilidade de mutação e vazio de geração *generation gap* – termo introduzido por De Jong para permitir sobreposição de populações.

Foram investigadas cinco variações da estratégia reprodutiva R1:

Estratégia reprodutiva R2 (*Elitist Model*) introduzida por De Jong, com a finalidade de manter os melhores cromossomas através da sua cópia para uma nova população. O elitismo força os algoritmos genéticos a reterem o mesmo número de melhores indivíduos em cada geração, a fim de que não sejam destruídos pelos operadores de cruzamento e mutação, ou perdidos se não forem seleccionados para reproduzir. O elitismo é uma versão artificial da selecção natural enunciada por Darwin, e conforme demonstrado pelo autor, melhora a procura local à custa de uma perspectiva global;

Estratégia reprodutiva R3 (*Expected Value Model*) a estratégia reprodutiva R3 foi criada para reduzir os erros estocásticos da selecção pelo giro da roleta, calculando uma probabilidade de selecção proporcional à aptidão e seleccionando os indivíduos de acordo com a referida distribuição de probabilidade. De Jong demonstrou que a estratégia R1 é susceptível a duas fontes de erro: (1) as aptidões médias dos *schemata* actuais são calculadas por amostragem finita sequencial, não é prático calcular através de outra forma; (2) o método de selecção é um processo de alta variância com uma quantidade apropriada de dispersão entre os números de cópias esperado e actual. De Jong tentou reduzir o segundo erro com R3 calculando um número esperado de descendentes para cada string (n indivíduos numa população):

$$\frac{f_i}{\bar{f}} \times n$$

sendo que de cada vez que um cromossoma for seleccionado para cruzamento o seu valor esperado de descendentes reduz em 0.5; se o cromossoma for copiado para a população seguinte o seu valor esperado de descendentes reduz em 1.0. Em qualquer dos casos sempre que o valor esperado dos descendentes de um cromossoma seja menor ou igual a zero, o cromossoma deixa de ser seleccionado para gerar descendentes.

Estratégia reprodutiva R4 (*Elitist Expected-Value Model*) consiste na combinação das estratégias R2 e R3. A estratégia resultou em melhorias consideráveis para funções unimodais;

Estratégia reprodutiva R5 (*Crowding Factor Model*) difere da estratégia R1 principalmente porque o processo de substituição dos indivíduos é diferente; nesta estratégia, em lugar da substituição total em cada geração, quando um indivíduo é criado, um outro deve ser imediatamente eliminado;

Estratégia reprodutiva R6 (*Generalized Crossover Model*) nesta estratégia, o cromossoma é considerado como um anel e um número pré-determinado de pontos de cruzamento CP (*number of crossover points*) é seleccionado uniformemente ao redor do círculo. Portanto, existem $C_{CP,L}$ combinações (L: tamanho do cromossoma) passíveis de serem seleccionadas, e à medida que CP aumenta, cada combinação torna-se menos provável de ser escolhida durante um cruzamento particular. Noutras palavras, segundo as experiências de De Jong, se CP aumenta, aumentam também as misturas aleatórias de partes dos cromossomas como se o cruzamento fosse uma simples *troca* e consequentemente diminui a preservação dos *schemata* importantes.

3.12 Decisões para implementar um algoritmo genético

O sucesso de um algoritmo genético utilizado num problema que contenha as características descritas na secção 3.3, depende principalmente de quatro decisões descritas a seguir:

Tipo de codificação Esta decisão é fundamental para o sucesso dos algoritmos genéticos e segundo Mitchell (1996) ainda não há nenhuma conclusão rigorosa sobre qual o tipo de codificação que conduzirá ao melhor resultado. Em muitas aplicações, tem sido mais adequado utilizar alfabetos de poucos caracteres ou números reais em vez da codificação binária;

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

Tipo de operadores Esta decisão depende do problema abordado, bem como da estratégia de codificação do cromossoma e do tipo de relacionamento entre genes;

Ajuste de parâmetros Ainda não existem resultados conclusivos sobre o estabelecimento de parâmetros como o tamanho da população, a taxa de cruzamento e a taxa de mutação. As experiências de De Jong indicam que o melhor tamanho de população se situa entre 50-100 indivíduos, taxa de cruzamento em aproximadamente 0.6 e taxa de mutação em aproximadamente 0.001; Grefenstette (1986) obteve tamanho de população 30, taxa de cruzamento 0.95 e taxa de mutação 0.01; Schaffer *et al.* (1989) chegaram aos seguintes valores respectivamente: 20-30, 0.75-0.95 e 0.005-0.01.

3 A estrutura básica do algoritmo

A estrutura básica do algoritmo, expressa através da representação em pseudo-código, ver **fig. 3.2**:

fig. 3.2 Pseudo-código da estrutura básica de um algoritmo genético.

Algoritmo genético

Gerar população inicial {de *n* indivíduos}
Avaliar a aptidão de cada indivíduo

Enquanto a condição de paragem não for verdadeira
{
  Repetir até que a nova população esteja completa
  {
    Seleccionar dois progenitores da geração
    Cruzar para gerar dois descendentes
    Mutar os dois descendentes
    Avaliar aptidão dos dois descendentes
    Colocar os descendentes na nova população
  }
  Geração actual = Nova população
}

3.14 Conclusão

Um algoritmo genético trabalha com base na geração e na evolução contínua de populações de cromossomas. Cada estrutura de cromossoma codifica uma solução do problema e representa um ponto no espaço de procura. A sua aptidão está associada com o valor da função objectivo.

Os algoritmos genéticos constituem um instrumento novo de investigação, capaz de ser utilizado em diversas áreas das actividades humanas. A sua principal vantagem reside na diversidade genética: a cada passo do processo de procura, um conjunto de candidatos é considerado e envolvido na criação de novos candidatos. Existem outras vantagens importantes, mas apesar disto, a utilização desta técnica heurística deve estar sempre restrita aos problemas cujas características

sejam coerentes com o seu campo de aplicação. Em termos de comparação com outros métodos heurísticos, pode-se afirmar que apesar dos muitos estudos e experiências realizadas, estas ainda são insuficientes para estabelecer conclusões precisas e rigorosas, como sustenta Reeves (1993).

A obtenção de um bom resultado pelos algoritmos genéticos, dependerá da eficácia de detalhes tais como o método de codificação das soluções candidatas, dos tipos de operadores genéticos utilizados, dos valores dos parâmetros e outros critérios utilizados. Um algoritmo genético na sua forma básica pode ser insuficiente para um determinado tipo de problema, por exemplo: a codificação binária pode ser inadequada; a selecção através da proporcionalidade da aptidão pode conduzir à convergência prematura; os operadores de cruzamento e de mutação podem tornar não admissíveis as soluções descendentes, ou até destruir bons esquemas dos cromossomas actuais e uma inadequação dos parâmetros pode diminuir o desempenho global. Por este motivo, as decisões sobre cada passo da elaboração do algoritmo devem ser analisadas cuidadosamente, visto que existem muitas possibilidades de implementação e o relacionamento entre os objectos de decisão geralmente não são suficientemente claros e simples.

Finalmente, é de salientar que embora alguns elementos sejam gerados aleatoriamente, a procura realizada pelos algoritmos genéticos não é aleatória; ela procura explorar eficientemente a informação histórica. Segundo Reeves (1993), da perspectiva da Investigação Operacional, a ideia de um algoritmo genético pode ser entendida como o uso inteligente de uma procura aleatória.

04 ABORDAGEM PROPOSTA

.1 Introdução

Conforme descrito na revisão da literatura (Capítulo 2) os métodos exactos não são capazes de produzir soluções óptimas em tempo útil para problemas de grande dimensão do tipo RCPSP, RCMPSP e JSP. Por este motivo optou-se pela utilização de métodos aproximados para a resolução deste tipo de problemas.

.2 Geradores de planos

Os geradores de planos apresentados neste trabalho são constituídos pelas seguintes duas componentes:

- Esquemas geradores de planos;
- Meta-heurísticas para melhorar a qualidade dos planos obtidos pelos esquemas.

A nível da construção de planos foi desenvolvido um novo esquema de geração que permite gerar planos activos parametrizados. Este novo esquema de geração de planos activos parametrizados será genericamente descrito na secção 4.3 e detalhadamente para cada tipo de problema, nos capítulos 5, 6 e 7.

Para evoluir a qualidade dos planos foi desenvolvido um algoritmo genético baseado em chaves aleatórias que será descrito genericamente na secção 4.4 e detalhadamente para os casos específicos dos problemas RCPSP, RCMPSP e JSP nos capítulos 5, 6 e 7.

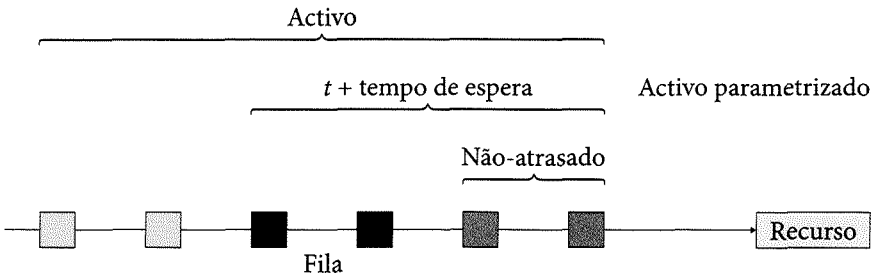
.3 Planos activos parametrizados

Dado que o conjunto dos planos activos é extremamente numeroso e que inclui um elevado número de soluções de má qualidade (devido aos grandes tempos de espera permitidos), optou-se por reduzir o número de planos activos considerados na procura, limitando esta apenas àqueles que não excedem determinado valor do tempo de espera e que serão designados por planos activos parametrizados.

Na geração de planos não-atrasados são consideradas disponíveis para planeamento no instante t todas as actividades que possam ser iniciadas nesse instante. No caso dos planos activos parametrizados foi incorporado um factor designado tempo de espera, que permite considerar como disponíveis para planeamen-

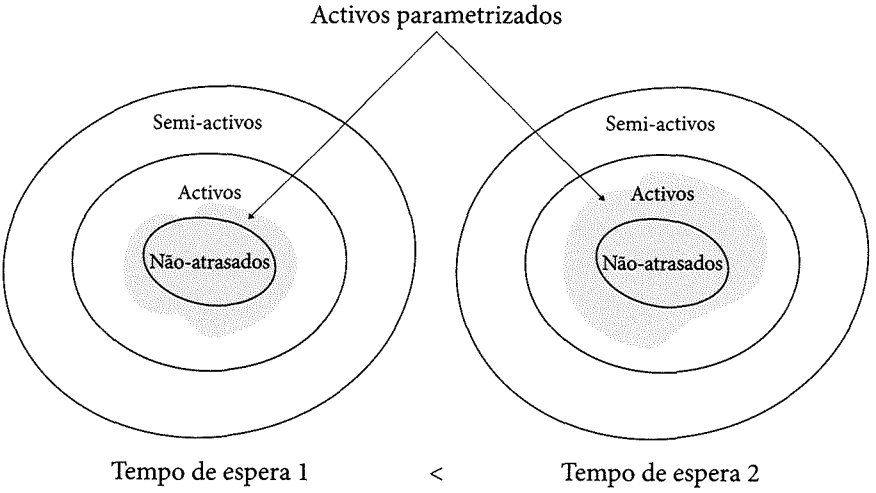
to no instante t, todas as operações que possam ser iniciadas até t + tempo de espera, ver **fig. 4.1**.

fig. 4.1 Actividades disponíveis para planeamento no instante t para o caso dos planos activos parametrizados.



À variação do tempo de espera corresponde uma diminuição ou aumento do espaço de soluções. Um valor nulo para o tempo de espera significa que o espaço de soluções está restrito aos planos não-atrasados. A **fig. 4.2** mostra onde se situa, relativamente às classes de planos semi-activos, activos e não-atrasados, o espaço de soluções que o gerador de planos activos parametrizados permite gerar.

fig. 4.2 Espaço de soluções relativo aos planos activos parametrizados.



contém a solução óptima. Com vista a reduzir esta possibilidade optou-se por escolher tempos de espera máximos relativamente elevados, isto é, 1.5 × duração da actividade mais longa.

4 Algoritmos genéticos baseados em chaves aleatórias

Com vista a melhorar a qualidade dos planos gerados pelo gerador de planos activos parametrizados, foi desenvolvido um algoritmo genético que usa chaves aleatórias como método de representação de soluções em vez da tradicional codificação binária. Esta metodologia foi proposta por Bean (1994) e consiste na atribuição de números aleatórios, compreendidos entre 0 e 1, aos genes que constituem um cromossoma.

4.4.1 Estratégia evolutiva

Inicialmente é gerada aleatoriamente uma população de cromossomas. Esta população será posteriormente transformada através da aplicação dos operadores genéticos.

A reprodução é inicialmente conseguida usando uma estratégia elitista, Goldberg (1989) na qual os melhores indivíduos de uma população são copiados para a geração seguinte. A vantagem desta estratégia é garantir que a melhor solução vá melhorando sucessivamente de geração para geração.

O operador de cruzamento adoptado é do tipo uniforme parametrizado, Spears e DeJong (1991), em vez dos tradicionais cruzamentos simples, duplo ou multiponto. Neste tipo de cruzamento são escolhidos aleatoriamente dois cromossomas da população (incluindo os cromossomas que foram seleccionados para pertencerem à geração seguinte) para darem origem a um cromossoma filho, e para cada gene do cromossoma filho é determinado qual o progenitor que contribuirá para o valor. É definida uma probabilidade de escolha, e para cada gene é gerado um número aleatório entre 0 e 1. Se o número gerado for inferior ou igual à probabilidade de escolha, é escolhido o gene do primeiro progenitor, caso contrário será escolhido o do segundo progenitor. Suponhamos que a probabilidade de escolha é 0.7, a **fig. 4.3** exemplifica o processo de cruzamento de dois cromossomas:

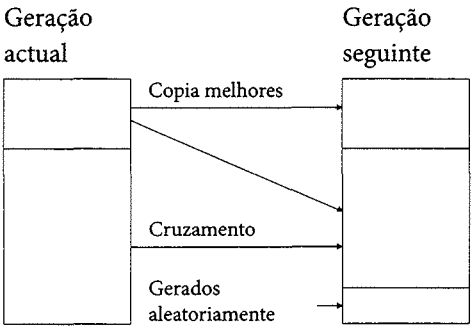
Cromossoma Progenitor 1	0.30	0.79	0.55	0.87
Cromossoma Progenitor 2	0.26	0.12	0.91	0.44
Valores gerados	0.58	0.89	0.52	0.25
	< 0.7	> 0.7	< 0.7	< 0.7
Cromossoma Filho	0.30	0.12	0.55	0.87

Se observamos o exemplo da figura anterior verificamos que o primeiro gene do cromossoma filho (0.30) foi contribuído pelo cromossoma do primeiro progenitor. Isto deve-se ao facto de o valor gerado para o primeiro gene (0.58) ser inferior à probabilidade de escolha (0.7) definida.

O operador de mutação em vez de utilizar a mutação gene a gene com baixa probabilidade, emprega o conceito de imigração. Em cada geração são gerados aleatoriamente novos membros a partir da mesma distribuição utilizada para a geração da população inicial. Este processo visa evitar a convergência prematura de uma população.

A **fig. 4.4** exemplifica o processo de transição de uma geração para a seguinte:

fig. 4.4 Transição de geração.



Introdução

Um dos objectivos mais comuns na gestão de projectos é a minimização da duração total de um projecto. A minimização da duração de um projecto pode trazer vantagens, por exemplo, nos seguintes aspectos:

- Permite libertar mais rapidamente recursos para serem utilizados em futuros projectos;
- Reduz o risco do não cumprimentos dos prazos estabelecidos.

Esta medida é geralmente utilizada sempre que o objectivo é planear apenas um projecto.

Apresenta-se neste capítulo uma nova abordagem para o problema do planeamento de um projecto com recursos limitados e no qual o objectivo é minimizar a duração de um projecto (*makespan*).

5.2 Formulação do problema

O problema do planeamento operacional de um projecto com recursos limitados RCPSP pode ser descrito da seguinte forma: um projecto consiste num conjunto de actividades $J=\{0, 1, 2, ..., n, n+1\}$ a programar. As actividades 0 e $n+1$ são fictícias, têm duração nula e correspondem ao início e fim do projecto. As actividades estão interrelacionadas por dois tipos de restrições:

- Restrições de precedência que implicam que cada actividade j só pode ser programada após todas as suas actividades precedentes, P_j, terem sido concluídas.
- Restrições de capacidade que implicam que uma actividade só pode ser programada se existirem recursos disponíveis.

No problema RCPSP existem K tipos de recursos, representados pelo conjunto $K=\{1,...,k\}$. Cada actividade tem uma duração definida por d_j. Entre os instantes de início e de fim, cada actividade j necessita de $r_{j,k}$ de cada recurso k e não pode ser interrompida. Cada tipo de recurso k dispõe de uma capacidade limitada R_k em qualquer instante. Os parâmetros d_j, $r_{j,k}$ e R_k são determinísticos; para as actividades de início e fim do projecto $d_0=d_{n+1}=0$ e $r_{0,k}=r_{n+1,k}=0$ para qualquer recurso k.

disponível de cada recurso, por forma a que a duração total do projecto, *make-span*, seja minimizada.

Considere-se que F_j representa o tempo de fim da actividade j. Um plano pode ser representado por um vector de tempos de fim das actividades $(F_1, F_2, \ldots, F_n)$. O conjunto de actividades que estão activas (em processamento) em determinado instante t designa-se $A(t)$.

O modelo conceptual do problema RCPSP descrito por Christofides et al. (1987) é o seguinte:

$$\text{Min } F_{n+1} \tag{5.1}$$

Sujeito a:

$$F_l \leq F_j - d_j \qquad j = 1,\ldots,n+1 \; ; \; l \in P_j \tag{5.2}$$

$$\sum_{j \in A(t)} r_{j,k} \leq R_k \qquad k \in K \; ; \; t \geq 0 \tag{5.3}$$

$$F_j \geq 0 \qquad j = 1,\ldots,n+1 \tag{5.4}$$

A função objectivo (5.1) minimiza o tempo de fim da última actividade do projecto, designada pela actividade $n+1$, minimizando assim a duração total do projecto. As restrições (5.2) impõem as relações de precedência entre actividades e as restrições (5.3) impõem para cada recurso k que a utilização resultante das actividades activas em cada instante t não exceda a capacidade disponível. Finalmente (5.4) impõe que as variáveis de decisão sejam não negativas.

5.3 Nova abordagem

A nova abordagem apresentada nesta publicação combina um algoritmo genético com um procedimento que gera planos activos parametrizados. Em termos gerais a abordagem consiste nas seguintes duas fases:

Construção de planos Esta fase recorre a um esquema de geração de planos do tipo paralelo modificado para construir planos activos parametrizados. A escolha da actividade a seleccionar em cada iteração do algoritmo é controlada através da prioridade e do tempo de espera associado a cada actividade;

mo genético para evoluir (melhorar) as prioridades e tempo de ... das na fase anterior.

Nas secções seguintes serão apresentados os detalhes da abordagem.

1 Esquema de geração de planos activos parametrizados

O procedimento utilizado para construir planos activos parametrizados combina a lógica dos esquemas série e paralelo descritos no capítulo 2.

Com vista a poder definir formalmente o algoritmo apresenta-se seguidamente alguma notação. Para cada iteração g, existe um tempo t_g que lhe está associado. O conjunto activo de actividades compreende o conjunto de todas as actividades que estão activas em t_g, i.e.

$$A_g = \left\{ j \in J \mid F_j - d_j \leq t_g < F_j \right\}.$$

A capacidade disponível do recurso k no instante t_g é dada por

$$RD_k\left(t_g\right) = R_k\left(t_g\right) - \sum_{j \in A_g} r_{j,k}.$$

S_g é o conjunto de todas as actividades programadas até à iteração g, e T_g representa os tempos de conclusão das actividades em S_g. Seja $tEspera_g$ o tempo de espera associado à iteração g e E_g o conjunto de todas as actividades que podem ser programadas (cujas predecessoras já foram programadas) no intervalo $[t_g, t_g + tEspera_g]$, i.e.

$$E_g = \left\{ j \in J \setminus S_{g-1} \mid F_i \leq t_g + tEspera_g \, (i \in P_j) \right\}.$$

A **fig. 5.1** descreve o pseudo-código do algoritmo utilizado para gerar planos activos parametrizados.

Inicialização: $g=1,\ t_1=0,\ A_0=\{0\},\ \mathcal{T}_0=\{0\},\ S_0=\{0\},\ RD_k(0)=R_k(k\in K)$

Enquanto $|S_g|<n+2$ **Repetir**

{

Actualizar E_g

Enquanto $E_g\neq\{\}$ **Repetir**

{

Seleccionar actividade com maior prioridade

$$j^* = \underset{j\in E_g}{\operatorname{argmax}} \left\{ PRIORIDADE_j \right\}$$

Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

$$FMC_{j^*} = \max_{i\in P_{j^*}} \{F_i\} + d_{j^*}$$

Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

$$F_{j^*} = \min\{\ t\in\left[FMC_{j^*} - d_{j^*}, \infty \right] \cap \mathcal{T}_g \mid\ r_{j^*,k} \le RD_k(\tau)$$

$$k\in K \mid r_{j^*,k}>0\ ,\ \tau\in\left[t,\ t+d_{j^*}\right]\ \}+d_{j^*}$$

Actualizar $S_g = S_{g-1}\cup\left\{ j^* \right\}$, $\mathcal{T}_g = \mathcal{T}_{g-1}\cup\left\{ F_{j^*} \right\}$

Incrementar a iteração: $g=g+1$

Actualizar $A_g, E_g, RD_k(t)\mid t\in\left[F_{j^*} - d_{j^*}, F_{j^*} \right], k\in K \mid r_{j^*,k}>0$

}

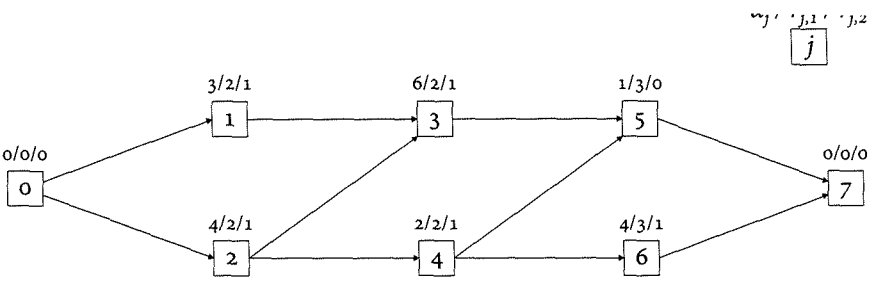
Determinar o tempo associado à iteração g

$t_g = \min\left\{ t\in\mathcal{T}_{g-1} \mid t>t_{g-1} \right\}$

}

5.3.1.1 Exemplo

Com vista a ilustrar o esquema de geração de planos activos parametrizados, apresenta-se a sua aplicação passo a passo ao projecto usado no capítulo 2 e que se reproduz seguidamente na **fig. 5.2**:



A **tab. 5.1** apresenta os valores das prioridades e dos tempos de espera a serem utilizados no exemplo que ilustra o esquema de geração de planos activos parametrizados.

tab. 5.1 Prioridades e tempos de espera usados no exemplo.

Actividade / Iteração j/g	$PRIORIDADE_j$	$tEspera_g$
0	0.00	0.00
1	0.55	1.98
2	0.61	2.31
3	0.35	3.96
4	0.52	4.13
5	0.08	8.05
6	0.27	7.25
7	0.00	0.00

- **Inicialização / Iteração $g = 0$**

$t_1 = 0$, $A_0 = \{0\}$, $\mathcal{F}_0 = \{0\}$, $S_0 = \{0\}$,

$RD_k(t)$=

k \ t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	>15
1	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Calcular $E_1 = \{j \in J \setminus S_0 | F_i \leq 0 + 1.98(i \in P_j)\} = \{1, 2\}$

Seleccionar actividade com maior prioridade

$$j^* = \underset{j \in E_1}{\operatorname{argmax}} \{0.55, 0.61\} = 2$$

Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

$FMC_2 = 0 + 4 = 4$

Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

$F_2 = 0 + 4 = 4$

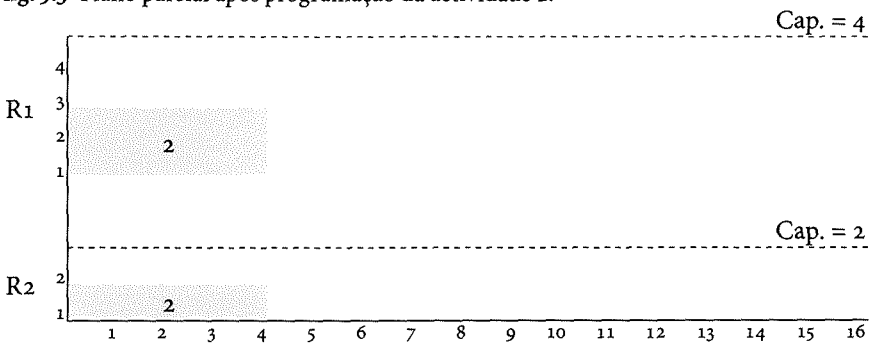
$S_1 = S_0 \cup \{2\} = \{0\} \cup \{2\} = \{0, 2\}$

$\mathcal{F}_1 = \mathcal{F}_0 \cup \{4\} = \{0\} \cup \{4\} = \{0, 4\}$

$\boldsymbol{g = 1 + 1 = 2}$

O plano parcial do esquema de geração de planos activos parametrizados após a iteração $g = 1$ pode ser visualizado na **fig. 5.3**.

fig. 5.3 Plano parcial após programação da actividade 2.



Actualizar $A_2 = \{0,2\}$

Actualizar $E_2 = \{ j \in J \setminus S_1 | F_i \leq 0 + 2.31(i \in P_j) \} = \{1\}$

$RD_k(t)=$

k \ t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	>15
1	2	2	2	2	4	4	4	4	4	4	4	4	4	4	4	4
2	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2

Seleccionar actividade com maior prioridade

$$j^* = \underset{j \in E_2}{\operatorname{argmax}} \{0.55\} = 1$$

Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

$FMC_1 = 0 + 3 = 3$

Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

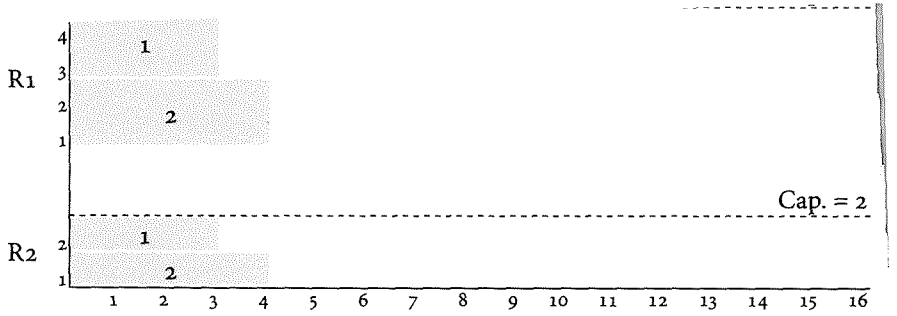
$F_1 = 0 + 3 = 3$

$S_2 = S_1 \cup \{1\} = \{0,2\} \cup \{1\} = \{0,1,2\}$

$\mathcal{F}_2 = \mathcal{F}_1 \cup \{3\} = \{0,4\} \cup \{3\} = \{0,3,4\}$

$\boldsymbol{g = 2 + 1 = 3}$

O plano parcial do esquema de geração de planos activos parametrizados após a iteração $\boldsymbol{g = 2}$ pode ser visualizado na **fig. 5.4**:



- **Iteração $g=3$**

  Actualizar $A_3=\{0,1,2\}$

  Actualizar $E_3=\{\, j\in J\setminus S_2 \mid F_i\le 0+3.96\,(i\in P_j)\}=\{\ \}$

  $RD_k(t)=$

k \ t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	>15
1	0	0	0	2	4	4	4	4	4	4	4	4	4	4	4	4
2	0	0	0	1	2	2	2	2	2	2	2	2	2	2	2	2

  Determinar o tempo associado à iteração $g=3$

  $t_3=\min\{3,4\}=3$

  Actualizar $E_3=\{\, j\in J\setminus S_2 \mid F_i\le 3+3.96\,(i\in P_j)\}=\{\ 3,\ 4\ \}$

  Seleccionar actividade com maior prioridade

  $$j^*=\underset{j\in E_3}{\operatorname{argmax}}\{\ 0.35\ ,\ 0.52\ \}=4$$

  Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

  $FMC_4=4+2=6$

  Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

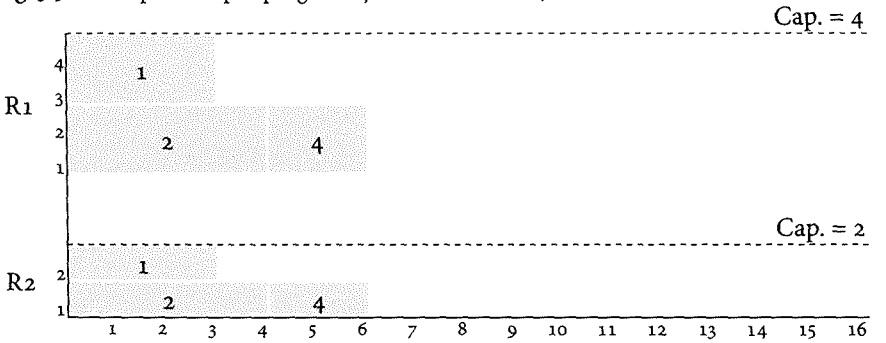
$S_3 = S_2 \cup \{4\} = \{0,1,2\} \cup \{4\} = \{0,1,2,4\}$

$T_3 = T_2 \cup \{6\} = \{0,3,4\} \cup \{6\} = \{0,3,4,6\}$

$g = 3 + 1 = 4$

O plano parcial do esquema de geração de planos activos parametrizados após a iteração $g = 3$ pode ser visualizado na **fig. 5.5**:

fig. 5.5 Plano parcial após programação da actividade 4.



- **Iteração** $g = 4$

Actualizar $A_4 = \{2,4\}$

Actualizar $E_4 = \{ j \in J \setminus S_3 \mid F_i \leq 3 + 4.13 \, (i \in P_j) \} = \{3,6\}$

$RD_k(t) =$

k \ t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	>15
1	0	0	0	2	2	2	4	4	4	4	4	4	4	4	4	4
2	0	0	0	1	1	1	2	2	2	2	2	2	2	2	2	2

Seleccionar actividade com maior prioridade

$$j^* = \underset{j \in E_4}{\operatorname{argmax}} \{0.35, 0.27\} = 3$$

$FMC_3 = 4 + 6 = 10$

Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

$F_3 = 4 + 6 = 10$

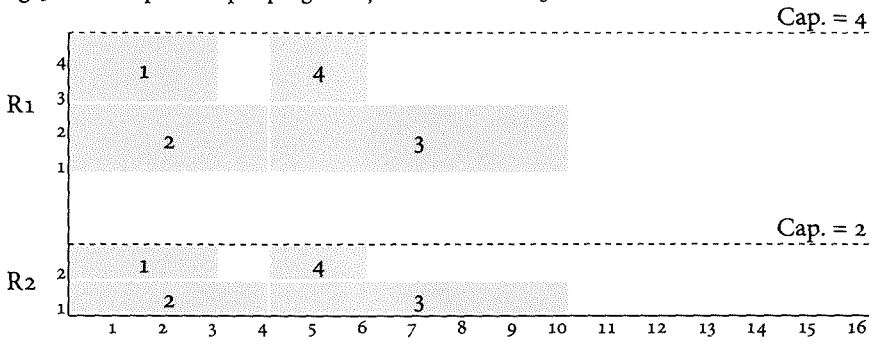
$S_4 = S_3 \cup \{3\} = \{0,1,2,4\} \cup \{3\} = \{0,1,2,3,4\}$

$T_4 = T_3 \cup \{10\} = \{0,3,4,6\} \cup \{10\} = \{0,3,4,6,10\}$

$\boldsymbol{g = 4 + 1 = 5}$

O plano parcial do esquema de geração de planos activos parametrizados após a iteração $g = 4$ pode ser visualizado na **fig. 5.6**:

fig. 5.6 Plano parcial após programação da actividade 3.



- **Iteração $g = 5$**

  Actualizar $A_5 = \{2,3,4\}$

  Actualizar $E_5 = \{ j \in J \setminus S_4 \mid F_i \leq 3 + 8.05\,(i \in P_j) \} = \{5,6\}$

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
k	1	0	0	0	2	0	0	2	2	2	2	4	4	4	4	4	4
	2	0	0	0	1	0	0	1	1	1	1	2	2	2	2	2	2

Seleccionar actividade com maior prioridade

$$j^* = \underset{j \in E_5}{\operatorname{argmax}} \{0.08, 0.27\} = 6$$

Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

$$FMC_6 = 6 + 4 = 10$$

Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

$$F_6 = 10 + 4 = 14$$

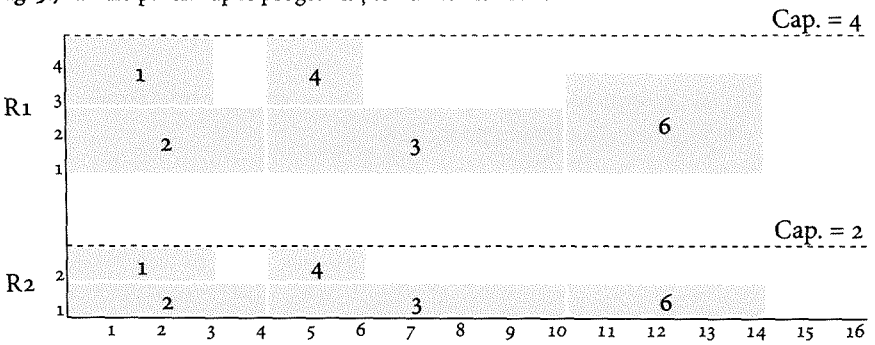
$$S_5 = S_4 \cup \{6\} = \{0,1,2,3,4\} \cup \{6\} = \{0,1,2,3,4,6\}$$

$$T_5 = T_4 \cup \{14\} = \{0,3,4,6,10\} \cup \{14\} = \{0,3,4,6,10,14\}$$

$$\boldsymbol{g = 5 + 1 = 6}$$

O plano parcial do esquema de geração de planos activos parametrizados após a iteração $g = 5$ pode ser visualizado na **fig. 5.7**:

fig. 5.7 Plano parcial após programação da actividade 6.



Actualizar $A_6 = \{2,3,4,6\}$

Actualizar $E_6 = \{ j \in J \setminus S_5 \mid F_i \leq 3 + 7.25\ (i \in P_j) \} = \{5\}$

$RD_k(t)=$

k	t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	>15
	1	0	0	0	2	0	0	2	2	2	2	1	1	1	1	4	4
	2	0	0	0	1	0	0	1	1	1	1	1	1	1	1	2	2

Seleccionar actividade com maior prioridade

$$j^* = \underset{j \in E_6}{\operatorname{argmax}} \{0.08\} = 5$$

Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

$FMC_5 = 10 + 1 = 11$

Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

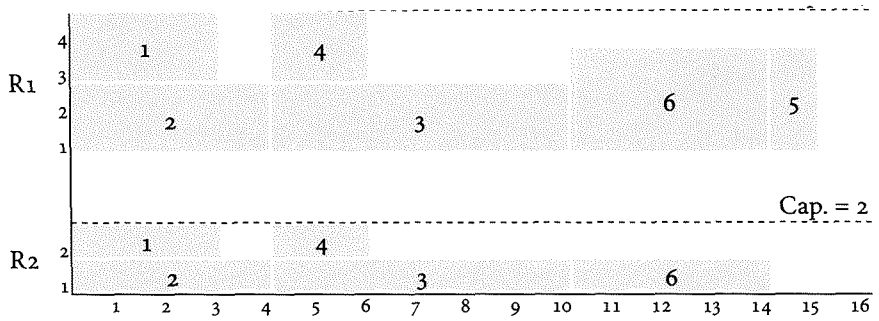
$F_5 = 14 + 1 = 15$

$S_6 = S_5 \cup \{5\} = \{0,1,2,3,4,6\} \cup \{5\} = \{0,1,2,3,4,5,6\}$

$T_6 = T_5 \cup \{15\} = \{0,3,4,6,10,14\} \cup \{15\} = \{0,3,4,6,10,14,15\}$

$\boldsymbol{g = 6 + 1 = 7}$

O plano parcial do esquema de geração de planos activos parametrizados após a iteração $\boldsymbol{g=6}$ pode ser visualizado na **fig. 5.8**:



- **Iteração** $g=7$

  Actualizar $A_7=\{2,3,4,5,6\}$

  Actualizar $E_7=\{j\in J\setminus S_6 \mid F_i\leq 3+0(i\in P_j)\}=\{\ \}$

  Determinar o tempo associado à iteração $g=7$ até $E_g\neq\{\}$

  $t_7=\min\{4,6,10,14,15\}=4$, $E_7=\{j\in J\setminus S_6 \mid F_i\leq 4+0\,(i\in P_j)\}=\{\ \}$

  $t_7=\min\{6,10,14,15\}=6$, $E_7=\{j\in J\setminus S_6 \mid F_i\leq 6+0\,(i\in P_j)\}=\{\ \}$

  $t_7=\min\{10,14,15\}=10$, $E_7=\{j\in J\setminus S_6 \mid F_i\leq 10+0\,(i\in P_j)\}=\{\ \}$

  $t_7=\min\{14,15\}=14$, $E_7=\{j\in J\setminus S_6 \mid F_i\leq 14+0\,(i\in P_j)\}=\{\ \}$

  $t_7=\min\{15\}=15$, $E_7=\{j\in J\setminus S_6 \mid F_i\leq 15+0\,(i\in P_j)\}=\{7\}$

  Seleccionar actividade com maior prioridade

  $j^*=\underset{j\in E_7}{\operatorname{argmax}}\{\ 0.00\ \}=7$

  Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

  $FMC_7=15+0=15$

$F_7 = 15 + 0 = 15$

$S_7 = S_6 \cup \{7\} = \{0,1,2,3,4,5,6\} \cup \{7\} = \{0,1,2,3,4,5,6,7\}$

$\mathcal{F}_7 = \mathcal{F}_6 \cup \{15\} = \{0,3,4,6,10,14,15\} \cup \{15\} = \{0,3,4,6,10,14,15\}$

$\boldsymbol{g = 7 + 1 = 8}$

Após esta iteração o plano final está concluído pois a condição é verdadeira. Dado que a actividade 7 (n+1) não tem duração nem ocupa recursos, o plano final é igual ao plano representado na **fig. 5.8**.

5.3.2 Algoritmo genético

Seguidamente descrevem-se os detalhes do algoritmo genético utilizado.

5.3.2.1 Representação cromossómica

Cada cromossoma (solução) é composto por $2n$ genes onde n representa o número de actividades. Os primeiros n genes são usados para obter as prioridades associadas a cada actividade e os últimos n genes são usados para obter os tempos de espera associados a cada iteração, conforme ilustra a figura seguinte.

$$\text{Cromossoma} = (\underbrace{\text{gene}_1, \ldots, \text{gene}_n}_{\text{Prioridades}}, \underbrace{\text{gene}_{n+1}, \ldots, \text{gene}_{2n}}_{\text{Tempos de espera}})$$

5.3.2.2 Descodificação das prioridades das actividades

Conforme mencionado anteriormente os primeiros n genes são associados a cada actividade. Inicialmente foi utilizada a seguinte expressão para descodificação das prioridades associadas a cada actividade:

$$PRIORIDADE_j = Gene_j.$$

tidos por esta descodificação, procurou melhorar-se o processo de descodificação, tentando incorporar neste alguma informação sobre a estrutura do problema.

A nova expressão descodificadora combina os seguintes dois factores:

Prioridade "ideal" prioridade que deveria ser usada caso os recursos tivessem capacidade infinita;

Factor de correcção factor que corrige a prioridade ideal de forma a ter em conta que a capacidade dos recursos é limitada.

Para determinar a prioridade ideal foi utilizada a seguinte expressão:

$$\frac{DCML_j}{DCC}$$

onde *DCMLj* representa a duração do caminho mais longo, i.e., com maior duração desde o início da actividade *j* até ao final do projecto e *DCC* representa a duração do caminho crítico do projecto, isto é, a duração do caminho com maior duração que tem início na primeira actividade e fim na última actividade. De notar que

$$0 \leq \frac{DCML_j}{DCC} \leq 1 .$$

Por exemplo para a actividade 4 da rede representada na **fig. 5.2** teriamos:

$$\frac{DCML_4}{DCC} = \frac{Max\{6,3\}}{11} = \frac{6}{11} .$$

O factor de correcção que ajusta a prioridade ideal de forma a ter em conta que a capacidade dos recursos é limitada é dado por:

$$\frac{1+gene_j}{2} .$$

Quando o valor do gene *j* é igual a 1 não existe qualquer correcção, quando o valor do gene *j* é igual a 0 o valor da prioridade é reduzido em 50%. Em resumo poderemos dizer que o factor de correcção poderá provocar uma redução até 50% do valor da prioridade ideal.

A expressão final de descodificação será então dada por:

$$PRIORIDADE_j = \frac{DCML_j}{DCC} \times \left[\frac{1+gene_j}{2}\right] \qquad j = 1,\ldots,n$$

Conforme se mencionou anteriormente os genes de n+1 até $2n$ são usados para obter os tempos de espera associados a cada iteração. O tempo de espera, *tEspera*$_g$, usado para realizar a programação na iteração g, é determinado pela seguinte expressão:

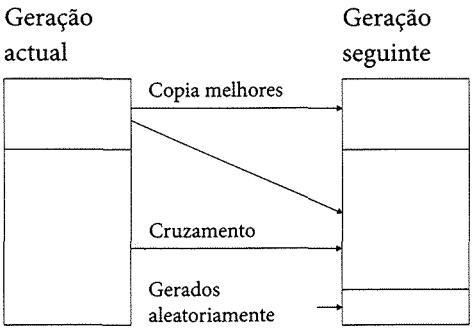
$$tEspera_g = Gene_{n+g} \times 1.5 \times MaxTempoProc \qquad g = 1, \ldots, n$$

na qual *MaxTempoProc* é o maior tempo de processamento de entre todas as actividades do projecto.

5.3.2.4 Estratégia evolutiva

A estratégia evolutiva segue o esquema descrito na secção 4.5.1, cujo esquema geral reproduzimos na **fig. 5.9**, existindo algumas alterações a nível da geração da população inicial e do processo de mutação/imigração.

fig. 5.9 Transição de geração.



Com vista a garantir a diversidade de planos incluídos nas populações tanto no processo de inicialização (primeira geração) como no processo de mutação/imigração usado para criar novas populações, impôs-se que 25% dos planos da população inicial e resultantes do processo de mutação/imigração fossem do tipo não-atrasado, i.e, com tempos de espera associados a todas as iterações iguais a zero, ou seja,

$$tEspera_g = 0 \qquad g = 1, \ldots, n.$$

testes experimentais, sendo este o valor que obteve melhor desempenho de entre os valores 0% a 100% com intervalos de 5%.

5.4 Testes experimentais

Esta secção apresenta os resultados dos testes experimentais obtidos pelo algoritmo GAPS (*Genetic Algorithm for Project Scheduling*).

Os testes experimentais foram realizados sobre um conjunto de problemas designados por J30, J60 e J120. Os conjuntos J30 e J60 consistem cada em 480 problemas e o conjunto J120 consiste em 600 problemas. Cada problema é constituído por um projecto que tem 30, 60 e 120 actividades respectivamente para os conjuntos de problemas J30, J60 e J120. Todos os problemas requerem 4 tipos de recursos. Detalhes dos problemas são descritos em Kolisch et al. (1998).

Para a realização dos testes experimentais o algoritmo genético usou a seguinte configuração para todos os problemas:

Tamanho da população	2 × número de actividades do problema
Cruzamento	probabilidade de cruzamento = 0.7
Selecção	10% dos melhores cromossomas são copiados para a próxima geração
Mutação	20% dos cromossomas da população
Mérito	*makespan* (a minimizar)
Sementes	10
Critério de paragem	1000 Gerações.

As variantes do algoritmo GAPS descritas em §5.3.2.2 serão designadas por:

GAPS-I O valor das prioridades das actividades é determinado utilizando a seguinte expressão:

$$PRIORIDADE_j = Gene_j$$

GAPS-II O valor das prioridades das actividades é determinado utilizando a seguinte expressão:

$$PRIORIDADE_j = \frac{DCML_j}{DCC} \times \left\lceil \frac{1+gene_j}{2} \right\rceil \qquad j = 1,\ldots,n$$

25% da população dos cromossomas da população inicial.

O algoritmo proposto é comparado com os seguintes algoritmos:

Problem and Heuristic Space
Leon, Ramamoorthy (1995)
Priority Rule Based Sampling Methods
Schirmer (1998)
Kolisch, Drexl (1996)
Kolisch (1996) – *single pass* LFT (*serial*)
Kolisch (1996) – *single pass* LFT (*parallel*)
Kolisch (1996) – *single pass* WCS
Kolisch (1995) – *random* (*serial*)
Kolisch (1995) – *random* (*paralell*)
Algoritmos Genéticos
Hartmann (2002) – GA-*self adapting*
Hartmann (1998) – GA-*activity list*
Hartmann (1998) – GA-*random key*
Hartmann (1998) – GA-*priority rule*
Simulate Annealing
Bouleimen, Lecocq (2003)
Tabu Search
Baar (1998).

Na **tab. 5.2** apresentam-se os resultados das três versões do algoritmo GAPS e dos algoritmos referenciados em Hartmann e Kolisch (2000). Para os problemas do conjunto J30 a medida utilizada foi o desvio médio percentual do makespan obtido em relação ao respectivo valor óptimo (D_{OPT}). Para os problemas dos conjuntos J60 e J120 foram utilizados o desvio médio percentual em relação à melhor solução obtida pelas outras heurísticas (D_{MS}) e o desvio médio percentual em relação a um limite inferior (D_{LI}) calculado por Hartmann e Kolisch (2000).

No conjunto J30 o algoritmo GAPS-III obteve $D_{OPT}=0.06$, sendo de realçar que este valor é inferior em cerca de 73% relativamente ao valor $D_{OPT}=0.22$ obtido pelo algoritmo que mais se aproxima, GA-*self adapting*.

Para os problemas do conjunto J60, o algoritmo GAPS-III obteve $D_{MS}=-0.08$ e $D_{LI}=11.16$. De notar que estes valores são inferiores em 124% e em 5% respectiva-

tab. 5.2 Resultados experimentais dos desvios médios percentuais.

Algoritmo	SGS	Autores	J30	J60		J120	
			D_{OPT}	D_{LI}	D_{MS}	D_{LI}	D
GAPS-III	Paralelo Mod.		0.06	11.16	−0.08	33.83	−1
GAPS-II	Paralelo Mod.		0.07	11.17	−0.07	34.31	−0
GAPS-I	Paralelo Mod.		0.10	11.62	0.26	36.10	0.
GA – *self adapting*	Série/Paralelo	Hartmann (2002)	0.22	11.70	—	35.39	-
GA – *activity list*	Série	Hartmann (1998)	0.25	11.89	0.33	36.74	0.
SA – *activity list*	Série	Bouleimen, Lecocq (2003)	0.23	11.90	0.37	37.68	1.
Sampling – adaptative	Série/Paralelo	Schirmer (1998)	0.44	12.59	0.82	38.70	2.
TS – *schedule scheme*	Heuristica Especial	Baar (1998)	0.44	13.48	1.42	—	-
Sampling – adaptative	Série/Paralelo	Kolisch, Drexl (1996)	0.52	13.06	1.14	40.45	3.
Single pass/sampling – LFT	Série	Kolisch (1996)	0.53	13.53	1.43	38.75	3.
GA – *random key*	Série	Hartmann (1998)	0.56	13.32	1.34	42.25	4.
Sampling – random	Série	Kolisch (1995)	1.00	15.17	2.73	47.61	8.
GA – *priority rule*	Série	Hartmann (1998)	1.12	12.74	0.94	38.49	1.
Single pass/sampling – WCS	Paralelo	Kolisch (1996)	1.28	13.21	1.44	38.77	2.
Single pass/ sampling – LFT	Paralelo	Kolisch (1996)	1.29	13.23	1.45	41.84	2.
Sampling – random	Paralelo	Kolisch (1995)	1.48	14.30	2.28	43.05	5.
GA – *problem space*	Paralelo	Leon, Ramamoorthy (1995)	1.59	13.49	1.71	40.69	3.

Para os problemas do conjunto J120, o algoritmo GAPS-III obteve $D_{MS} = -1.04$ e $D_{LI} = 33.83$. De notar que estes valores são inferiores em 255% e em 4% respectivamente em relação aos valores $D_{MS} = 0.67$ e $D_{LI} = 35.39$, obtidos pelos algoritmos que mais se aproximam GA-*activity list* e GA-*self adapting*, respectivamente.

Com vista a avaliar o desempenho em termos absolutos para os problemas dos conjuntos J60 e J120 foi utilizado o desvio médio percentual do *makespan* em relação ao melhor valor conhecido deste (D_{MS}), sendo o melhor valor obtido pela escolha dos melhores resultados obtidos por um grande número de autores, ver Apêndice B.

tab. 5.3 Resultados experimentais dos desvios médios percentuais em relação ao makespan.

Algoritmo	J60 D_{MS}	J120 D_{MS}
GAPS-III	0.65	3.54
GAPS-II	0.31	2.25
GAPS-I	0.30	1.86
Hartmann e Kolisch (2000)	0.38	2.94

Para os problemas do conjunto J60 o algoritmo GAPS-III apresenta os melhores resultados, tendo obtido $D_{MS} = 0.30$. De notar que este valor é inferior em 21% em relação ao valor de $D_{MS} = 0.38$, obtido pelos melhores valores de Hartmann e Kolisch (2000).

Para os problemas do conjunto J120 o algoritmo GAPS-III obteve os melhores resultados, tendo obtido $D_{MS} = 1.86$. De notar que este valor é inferior em 37% em relação ao valor $D_{MS} = 2.94$, obtido pelos melhores valores de Hartmann e Kolisch (2000).

Analisa-se de seguida a distribuição dos desvios médios percentuais em relação ao melhor valor conhecido do *makespan*, relativamente aos problemas dos conjuntos J30, J60 e J120. A **tab. 5.4** apresenta a distribuição dos desvios médios percentuais relativamente aos problemas do conjunto J30. O algoritmo GAPS-I obteve 451 (94%) problemas com valor óptimo $D_{MS} = 0$, o algoritmo GAPS-II obteve 463 (96%) problemas com valor óptimo e o algoritmo GAPS-III obteve 464 (97%) problemas com valor óptimo.

Desvios	Frequências absolutas		
	GAPS-I	GAPS-II	GAPS-III
0%	451	463	464
1%	1	0	0
2%	24	12	11
3%	4	3	3
4%	0	1	1
5%	0	1	1
> 5%	0	0	0
Total	480	480	480

A **tab. 5.5** apresenta a distribuição dos desvios médios percentuais relativamente aos problemas do conjunto J60. O algoritmo GAPS-I obteve 356 (74%) problemas com valor óptimo $D_{MS} = 0$, o algoritmo GAPS-II obteve 382 (80%) problemas com valor óptimo, o algoritmo GAPS-III obteve 388 (81%) problemas com valor óptimo e os melhores de Hartmann e Kolisch (2000) obtiveram 371 (77%) problemas com valor óptimo.

tab. 5.5 Distribuição dos desvios médios percentuais para os problemas do conjunto J60.

Desvios	Frequências absolutas			
	GAPS-I	GAPS-II	GAPS-III	Hartmann/Kolisch
0%	356	382	388	371
1%	6	27	21	22
2%	49	53	52	53
3%	31	13	13	25
4%	22	3	4	7
5%	14	1	2	2
6%	1	0	0	0
7%	1	0	0	0
8%	0	0	0	0
9%	0	0	0	0
10%	0	0	0	0
> 10%	0	0	0	0
Total	480	480	480	480

mas com $D_{MS}=0$, o algoritmo GAPS-II obteve 193 (32%), o algoritmo GAPS-III obteve 206 (34%) e os melhores de Hartmann e Kolisch (2000) obtiveram 181 (30%).

tab. 5.6 Distribuição dos desvios médios percentuais para os problemas do conjunto J120.

Desvios	Frequências absolutas			
	GAPS-I	GAPS-II	GAPS-III	Hartmann/Kolisch
0%	152	193	206	181
1%	12	20	41	12
2%	34	42	87	21
3%	39	121	106	67
4%	69	102	75	86
5%	80	85	53	114
6%	95	22	19	63
7%	73	12	10	30
8%	29	2	2	23
9%	14	0	1	3
10%	2	0	0	0
11%	1	0	0	0
> 10%	0	0	0	0
Total	600	600	600	600

O algoritmo foi implementado em Visual Basic 6.0 e os testes computacionais foram realizados num computador com sistema operativo Windows Me e um processador AMD a 1.33 GHz. Os tempos médios de computação em segundos para 1000 gerações por tipo de problema foram:

tab. 5.7 Tempo médio de computação por tipo de problema para 1000 Gerações.

Tipo de problema	J30	J60	J120
Tempo médio para 1000 gerações	8.78 s	34.78 s	164.35 s

5.5 Conclusões

Neste capítulo apresenta-se uma nova abordagem para o problema da geração de planos de um projecto com recursos limitados. Esta abordagem combina um algoritmo genético com um esquema de geração de planos.

recurso a prioridades e tempos de espera definidos pelo algoritmo genético. Os planos são obtidos através de um esquema de geração de planos activos parametrizados especificamente desenvolvido.

Os testes de desempenho do algoritmo realizados com base nos problemas dos conjuntos J30, J60 e J120 apresentados em Kolisch et al. (1998), demostram que os algoritmos GAPS-I, GAPS-II e GAPS-III obtêm, em termos relativos, um desempenho superior quando comparados com as heurísticas apresentadas em Hartmann e Kolisch (2000). Adicionalmente, para os problemas dos conjuntos J30, J60 e J120, os algoritmos GAPS-II e GAPS-III obtêm, em termos absolutos, também um desempenho superior quando comparados com os melhores valores conhecidos dos problemas dos conjuntos J30, J60 e J120 (de acordo com Hartmann e Kolisch, 2000).

6.1 Introdução

A crescente personalização de produtos e serviços tem tido como consequência a redução dos sistemas de produção do tipo repetitivo e o aumento dos sistemas de produção do tipo projecto, tornando assim o problema do planeamento de múltiplos projectos com recursos limitados cada vez mais relevante.

Apresenta-se neste capítulo uma nova abordagem para o problema do planeamento de múltiplos projectos com recursos limitados.

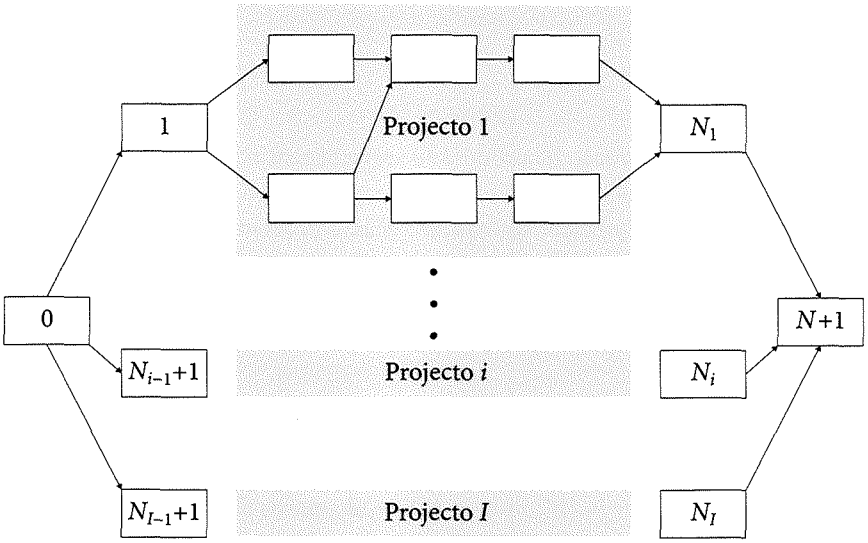
6.2 Formulação do problema

O problema consiste num conjunto de projectos I no qual cada projecto i é composto pelas actividades $j=\{N_{i-1}+1,\ldots,N_i\}$, sendo que as actividades $N_{i-1}+1$ e N_i são fictícias e representam o início e o fim do projecto i. As actividades estão interrelacionadas por dois tipos de restrições:

- Primeiro, as restrições de precedência que implicam que cada actividade $j \in J$ só pode ser programada após todas as suas actividades precedentes, P_j, terem sido concluídas;
- Segundo, as restrições de capacidade que implicam que uma actividade só pode ser programada se existirem recursos disponíveis.

No problema do RCMPSP existem K tipos de recursos, representados pelo conjunto $K=\{1,\ldots,k\}$. Cada actividade j tem uma duração definida por d_j e entre os instantes de início e de fim necessita de $r_{j,k}$ unidades de cada recurso k não podendo ser interrompida. Cada tipo de recurso dispõe em qualquer instante de uma capacidade limitada Rk. Os parâmetros d_j, $r_{j,k}$ e R_k são determinísticos. As actividades de início e fim de cada projecto $i \in I$, têm duração nula, i.e., $d_{(N_{i-1}+1)}=d_{N_i}=0$ e $r_{N_{i-1}+1,k}=r_{N_i,k}=0 (k \in K)$. As actividades 0 e $N+1$, correspondentes ao início e fim de todos os projectos, são fictícias e têm duração nula, ver **fig. 6.1**.

O problema do RCMPSP consiste em determinar os tempos de início e de fim de cada actividade respeitando as relações de precedência e a capacidade de cada recurso, de forma a minimizar determinada medida de desempenho.



Considere-se que F_j representa o tempo de fim da actividade j. Um plano pode-se representar por um vector $(F_1, F_2, \ldots, F_n)$. O conjunto de actividades que estão activas (em processamento) em determinado instante t designa-se $A(t)$.

O modelo conceptual do problema RCMPSP pode ser descrito da seguinte forma:

Minimizar *Medida de desempenho* $(F_j, j=1,\ldots,N)$ (6.1)

Sujeito a:

$$F_l \leq F_j - d_j \qquad j=1,\ldots,N+1 \; ; \; l \in P_j \tag{6.2}$$

$$\sum_{j \in A(t)} r_{j,k} \leq R_k \qquad k \in K \; ; \; t \geq 0 \tag{6.3}$$

$$F_j \geq 0 \qquad j=1,\ldots,N+1 \tag{6.4}$$

De referir que (6.1) representa a medida de desempenho a ser minimizada, (6.2) impõe as relações de precedência entre actividades e (6.3) impõe que a utilização do recurso k resultante das actividades em processamento no instante t não exceda a capacidade disponível. Finalmente (6.4) impõe que as variáveis de decisão não sejam negativas.

utilizados os custos associados aos em curso de fabrico e às existências.

6.3 Nova abordagem

A nova abordagem apresentada neste capítulo combina uma medida de desempenho inovadora ajustada à realidade prática do problema e uma metaheurística para resolução do problema. A metaheurística consiste num algoritmo genético e num procedimento que gera planos activos parametrizados. Em termos gerais a abordagem proposta é inovadora nos seguintes dois aspectos fundamentais:

Modelo É desenvolvida uma nova medida de desempenho ajustada à realidade prática integrando o cumprimento dos prazos de entrega, os em cursos de fabrico e as existências. É introduzido o conceito de data de libertação com vista a poder modelar outro tipo de restrições.

Método de resolução Dada a dificuldade de resolução de problemas reais por métodos exactos foi desenvolvido um novo método de resolução que combina um algoritmo genético com um esquema inovador que constrói planos activos parametrizados.

Nas secções seguintes são apresentados detalhes da abordagem.

6.3.1 Medida de desempenho

A gestão de projectos encontra-se actualmente perante um cenário de acrescida dificuldade, dado ter que:

- Entregar os produtos nas datas de entrega estabelecidas com os clientes;
- Não concluir os produtos antecipadamente para não incorrer em custos de armazenamento e em imobilização desnecessária de capital;
- Reduzir o tempo global de permanência dos produtos/serviços no sistema de forma a reduzir o valor dos em curso de fabrico e a descongestionar o sistema produtivo.

Perante o cenário atrás descrito, torna-se necessário ponderar os seguintes três critérios: atrasos, avanços e tempos de permanência no sistema. Com vista a poder ponderar todos estes critérios é necessário identificar claramente qual o custo que cada um implica para a empresa. Dado que entregar um produto com atraso

tempo de permanência estão relacionados com custos de capital pelo que são em geral ponderados de igual forma.

No modelo que a seguir se apresenta o critério tempo de permanência no sistema será substituído pelo que se designará desvio de fluxo e que representa a diferença entre o tempo total de permanência no sistema e o tempo produtivo. O tempo de permanência de um produto/serviço no sistema é constituído por duas componentes: os tempos produtivos e os tempos não produtivos (tempos gastos em fila de espera, transportes, etc.). Partindo do princípio que os tempos produtivos estão optimizados é apenas relevante a redução dos tempos não produtivos. A medida de desempenho desenvolvida incorpora em simultâneo os três critérios acima mencionados: atrasos, avanços e desvios de fluxo.

Com vista a poder definir formalmente a medida de desempenho será utilizada a seguinte notação:

D_i Duração ideal para o projecto i.
DE_i Data de entrega prevista para o projecto i.
DC_i Data de conclusão do projecto i no plano gerado.
DI_i Data de início do projecto i no plano gerado.
AT_i Atraso do projecto $i = max(DC_i - DE_i, 0)$.
AV_i Avanço do projecto $i = max(DE_i - DC_i, 0)$.
DF_i Desvio de fluxo do projecto $i = max(DC_i - DI_i - D_i, 0)$.
DCC_i Duração do caminho crítico do projecto$i$.

A nova medida de desempenho será definida pela seguinte função:

$$a\sum_i AT_i^3 + b\sum_i AV_i^2 + c\sum_i DF_i^2 \tag{6.5}$$

em que a, b e c são constantes às quais o agente de decisão, considerando a sua experiência, deverá atribuir valores de forma a ponderar cada um dos critérios.

Numa situação real não é conhecida a duração ideal de um projecto. Com vista a contornar este problema substituiu-se a função objectivo (6.5) por uma que possa conduzir a soluções de qualidade equivalente. A modificação efectuada consistiu em substituir o termo

$$c\sum_i DF_i^2 \quad \text{por} \quad c\sum_i \frac{(DC_i - DI_i)^2}{DCC_i}.$$

Na formulação apresentada em 6.2, as restrições relativas à utilização de recursos estão expressas nas condições (6.3). Em geral existem outros tipos de restrições relacionados com a data de início do projecto e que têm em conta recursos cuja utilização e disponibilidade são em geral avaliados de uma forma subjectiva e portanto não são de fácil incorporação nas expressões (6.3). Com vista a modelar este tipo de situação são adicionadas restrições que impõem que o projecto não possa ter início antes de determinada data. Este tipo de restrições é modelado da seguinte forma:

$$F_{N_{i-1}+1} \geq MDL_i \qquad i = 1,...,I$$

onde MDL_i designa a menor data de libertação do projecto i. Optou por incluir-se este tipo de restrição no modelo conceptual de forma implícita atribuindo uma duração DL_i à actividade inicial de cada projecto, i.e.,

$$d_{N_{i-1}+1} = DL_i \geq MDL_i \qquad i = 1,...,I\,.$$

6.3.3 Esquema de geração de planos activos parametrizados

O procedimento utilizado para construir planos activos parametrizados combina a lógica dos esquemas série e paralelo descritos no capítulo 2.

Com vista a poder definir formalmente o algoritmo adiciona-se seguidamente alguma notação. Para cada iteração g, existe um tempo tg que lhe está associado. O conjunto activo de actividades compreende o conjunto de todas as actividades que estão activas em t_g, i.e. $A_g = \{ j \in J \,|\, F_j - d_j \leq t_g < F_j \}$. A capacidade disponível do recurso k no instante t_g é dada por

$$RD_k(t_g) = R_k(t_g) - \sum_{j \in A_g} r_{j,k}\,.$$

S_g é o conjunto de todas as actividades programadas até à iteração g, e representa os tempos de conclusão das actividades em S_g. Seja $tEspera_g$ o tempo de espera associado à iteração g e E_g o conjunto de todas as actividades que podem ser programadas (cujas predecessoras já foram programadas) no intervalo $[\,t_g, t_g + tEspera_g]$, i.e.

$$E_g = \{ j \in J \setminus S_{g-1} \,|\, F_i \leq t_g + tEspera_g \ (i \in P_j) \}\,.$$

fig. 6.2 Pseudo-código para gerar planos activos parametrizados.

Inicialização: $g = 1, t_1 = 0, A_0 = \{0\}, \mathcal{T}_0 = \{0\}, S_0 = \{0\}, RD_k(0) = R_k \ (k \in K)$

Enquanto $|S_g| < n+2$ **Repetir**
{
  Actualizar E_g
  Enquanto $E_g \neq \{\}$ **Repetir**
  {
    Seleccionar actividade com maior prioridade

    $$j^* = \underset{j \in E_g}{\operatorname{argmax}} \left\{ PRIORIDADE_j \right\}$$

    Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

    $$FMC_{j^*} = \max_{i \in P_{j^*}} \{F_i\} + d_{j^*}$$

    Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

    $$F_{j^*} = \min \{ \ t \in \left[FMC_{j^*} - d_{j^*} \ , \infty \right] \cap \mathcal{T}_g \ \mid \ r_{j^*,k} \leq RD_k(\tau),$$
    $$k \in K \mid r_{j^*,k} > 0 \ , \ \tau \in \left[t, \ t \ + \ d_{j^*} \right] \ \} + d_{j^*}$$

    Actualizar $S_g = S_{g-1} \cup \left\{ j^* \right\}$, $\mathcal{T}_g = \mathcal{T}_{g-1} \cup \left\{ F_{j^*} \right\}$

    Incrementar a iteração: $\boldsymbol{g = g+1}$

    Actualizar $A_g, E_g, RD_k(t) \mid t \in \left[F_{j^*} - d_{j^*} \ , F_{j^*} \right], k \in K \mid r_{j^*,k} > 0$
  }
  Determinar o tempo associado à iteração $\boldsymbol{g}$
  $t_g = \min \left\{ t \in \mathcal{T}_{g-1} \mid t > t_{g-1} \right\}$
}

Seguidamente descrevem-se os detalhes do algoritmo genético utilizado.

6.3.4.1 Representação cromossómica

Cada cromossoma é composto por $2n+m$ genes onde n representa o número de actividades e m o número de projectos.

Os primeiros n genes são usados como prioridades das actividades $j=\{1,\ldots,n\}$. Os genes de $n+1$ até $2n$ são usados para determinar os tempos de espera associados a cada iteração $g=\{1,\ldots,n\}$ do procedimento gerador de planos activos parametrizados. Os genes de $2n+1$ até $2n+m$ são usados para determinar as datas de disponibilidade de cada projecto.

$$\text{Cromossoma} = (\underbrace{\text{gene}_1, \ldots, \text{gene}_n}_{\text{Prioridades}}, \underbrace{\text{gene}_{n+1}, \ldots, \text{gene}_{2n}}_{\text{Tempos de espera}}, \underbrace{\text{gene}_{2n+1}, \ldots, \text{gene}_{2n+m}}_{\text{Tempos de espera}})$$

6.3.4.2 Descodificação

Descrevem-se de seguida os procedimentos de descodificação dos parâmetros relativos às seguintes três alternativas:

GaGen procedimento de descodificação simples à qual não é adicionada qualquer informação específica do problema.

GaFolgaNA procedimento de descodificação simples à qual não é adicionada qualquer informação específica do problema, em que as prioridades das actividades são estáticas (i.e. não são objecto de evolução por parte do algoritmo genético) e em que os planos gerados são não-atrasados.

GaFolgaMod procedimento de descodificação à qual é adicionada informação específica do problema.

6.3.4.2.1 Descodificação das prioridades das actividades

As prioridades das actividades são determinadas a partir dos primeiros n genes de cada cromossoma. O valor da prioridade de uma actividade está compreendido entre 0 e 1. A sua forma de determinação varia de acordo com o algoritmo utilizado.

$$Prioridade_j = Gene_j$$

Para o algoritmo *GaFolgaNA* a prioridade é estática e incorpora a folga da actividade, sendo determinada de acordo com a seguinte expressão:

$$Prioridade_j = \frac{Folga_j}{MaiorFolga}$$

No algoritmo *GaFolgaMod* a prioridade incorpora o valor da folga da actividade de acordo com a seguinte expressão:

$$Prioridade_j = \frac{Folga_j}{MaiorFolga} \times \left(0.7 + 0.3 \times Gene_j\right)$$

onde

$DE_{i|j \in i}$ = Data de entrega do projecto i à qual pertence a actividade j.

$DCML_j$ = Duração do caminho mais longo desde o início da actividade j até ao fim do projecto à qual a actividade j pertence.

$Folga_j = DE_{i|j \in i} - DCML_j$

$MaiorFolga$ = Maior folga de entre todas as actividades de todos os projectos.

6.3.4.2.2 Descodificação dos tempos de espera

Conforme mencionado anteriormente os genes de $n+1$ até $2n$ são usados para obter os tempos de espera associados a cada iteração g. Para os algoritmos *GaGen* e *GaFolgaMod* o tempo de espera, $tEspera_g$, usado para realizar a programação na iteração g, é determinado pela seguinte expressão:

$$tEspera_g = Gene_{n+g} \times 1.5 \times MaxTempoProc \qquad g = 1 \ldots, n$$

na qual *MaxTempoProc* é o maior tempo de processamento de entre todas as actividades do projecto.

No algoritmo *GaFolgaNA* os tempos de espera associados a cada iteração g são nulos (planos não-atrasados).

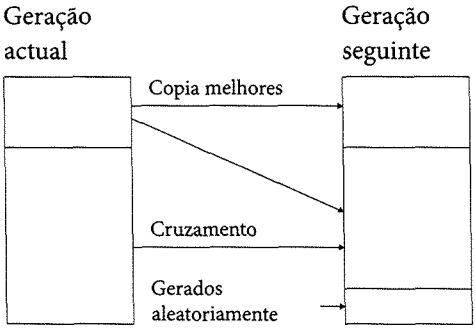
Os genes de $2n$ até $2n+m$ são usados para determinar as datas de libertação de cada projecto i. As alternativas *GaFolgaMod*, *GaGen* e *GaFolgaNA* utilizam a mesma expressão para a descodificação das datas de libertação de cada projecto i:

$$DL_i = MDL_i + Gene_{2n+i} \times \left(DE_i - MDL_i\right)$$

6.3.4.3 Estratégia evolutiva

A estratégia evolutiva segue o esquema descrito na secção 4.5.1, cujo esquema geral se reproduz na **fig. 6.3**, existindo algumas alterações a nível da geração da população inicial e do processo de mutação/imigração.

fig. 6.3 Transição de geração.



Com vista a garantir a diversidade de planos incluídos numa população tanto no processo de inicialização da população (primeira geração) como no processo de mutação/imigração usado para criar novas populações, impôs-se que 25% dos planos da população inicial e resultantes do processo de mutação/imigração fossem do tipo não-atrasado, i.e, com tempos de espera associados a todas as iterações iguais a zero, ou seja,

$$tEspera_g = 0 \qquad g = 1, \ldots, n.$$

O valor da percentagem igual a 25% de planos não-atrasados a incluir na população inicial e resultantes do processo de mutação/imigração foi obtido por

6.4 Gerador de problemas

Para o problema do planeamento de múltiplos projectos não existem na literatura problemas de referência disponíveis para utilizar nos testes experimentais pelo que se desenvolveu um gerador de problemas específico. Este gerador de problemas foi desenvolvido com o objectivo de gerar problemas próximos da realidade industrial.

Relativamente às características dos problemas que a literatura refere, salientam-se os seguintes trabalhos:

- Fendley (1968) utilizou problemas com 3 e 5 projectos.
- Kurtulus e Narula (1985) utilizaram problemas com 3 projectos, em que cada projecto tinha entre 24 e 33 actividades para problemas de pequena dimensão e entre 50 e 66 actividades para problemas de grande dimensão.
- Lova (1997) realizou um estudo na região de Valência em que procurou caracterizar as actividades, recursos e projectos de várias empresas de pequena e média dimensão. O estudo indicou que:

  · 84% das empresas trabalham em ambiente de múltiplos projectos (o que se aproxima do valor de 90% de empresas que trabalham em ambiente de múltiplos projectos obtido por Payne (1995));
  · 84% dos projectos tem menos de 50 actividades;
  · 95% dos projectos tem menos de 100 actividades.

- Lova et al. (2000) com base em Lova (1997), realizaram o seu trabalho com problemas de 4 e 8 projectos, cujas actividades variam entre 30 e 60 actividades. Desta forma, os problemas de pequena dimensão caracterizam-se por 120 actividades e os de grande dimensão por 480 actividades. Nestes problemas cada projecto tem em média 4 actividades em paralelo. Os problemas utilizaram, em média, 2 ou 4 recursos renováveis. As necessidades de capacidade de cada recurso foram geradas aleatoriamente entre 1 e 6.

Para poder avaliar o desempenho absoluto dos algoritmos havia necessidade de conhecer o valor óptimo da função desempenho associada a cada problema. Como esta não é conhecida contornou-se o problema impondo que os problemas

O gerador de problemas desenvolvido possui os seguintes parâmetros de entrada para permitir gerar vários tipos de problemas:

- Número de problemas a gerar;
- Número de projectos a incluir em cada problema;
- Número médio de projectos a serem executados em simultâneo.

Os projectos usados na construção dos problemas são todos do tipo J120 [Kolisch (1998)], cada um com 120 actividades e utilizando até 4 recursos.

Na construção de cada problema utilizaram-se as seguintes regras:

1 Os projectos a incluir em cada problema são escolhidos aleatoriamente de entre o conjunto de 600 projectos disponíveis do tipo J120.
2 O valor atribuído à duração ideal de cada um dos projectos incluídos no problema foi obtido a partir dos melhores valores conhecidos do *makespan*.
3 O número médio de projectos a executar em simultâneo foi imposto de forma indirecta obrigando a que todos os projectos fossem concluídos numa duração máxima do problema igual a

$$\frac{\textit{Somatório das durações ideais de cada projecto}}{\textit{Número de projectos em execução em simultâneo}}.$$

Caso a duração ideal de algum dos projectos exceda a duração máxima do problema será gerado um novo problema (Esta situação nunca aconteceu dado que o número de projectos incluídos em cada problema é substancialmente superior ao número de projectos em execução em simultâneo).
4 A data de início de cada projecto foi gerada aleatoriamente no intervalo [0, duração máxima do problema − duração ideal].
5 A capacidade de cada recurso ao longo do intervalo de tempo [0, duração máxima do problema] foi calculada somando a capacidade de cada recurso em cada projecto desde a data de início definida na regra 4 até à data de início mais a duração ideal de cada projecto obtida na regra 1. De notar que esta atribuição de capacidades aos recursos garante que o valor óptimo da medida de desempenho será igual a zero.

O pseudo-código do gerador dos problemas teste apresenta-se no Apêndice B.

Nesta secção são apresentados os resultados obtidos pelas alternativas *GaGen*, *GaFolgaNA* e *GaFolgaMod*.

Para a realização do estudo foram gerados problemas tipo com 10, 20, 30 e 50 projectos no qual cada projecto contém 120 actividades o que implica que cada tipo de problema tenha respectivamente 1200, 2400, 3600 e 6000 actividades. Para cada problema tipo foram geradas 20 instâncias e o número médio de projectos a serem executados em simultâneo foi 3, 6, 9 e 15, respectivamente para os problemas com 10, 20, 30 e 50 projectos.

Para a realização dos testes experimentais o algoritmo genético usou a seguinte configuração para todos os problemas:

Tamanho da população	Mínimo (0.2 × número de actividades, 250)
Cruzamento	Probabilidade de cruzamento = 0.7
Selecção	10% dos melhores cromossomas são copiados para a próxima geração
Mutação	20% dos cromossomas da população
Mérito	Foi usada a equação (6.5)
Sementes	1
Critério de paragem	50 gerações.

Na **tab. 6.1** apresentam-se os resultados relativos aos testes computacionais realizados. As colunas *Média*[1] e *Desvio Padrão*[1], *Média*[2] e *Desvio Padrão*[2], *Média*[3] e *Desvio Padrão*[3] e *Média*[4] e *Desvio Padrão*[4] representam respectivamente a média e o desvio padrão obtidos para as 20 instâncias dos valores obtidos pelas seguintes expressões:

$$\frac{a\sum_{i=1}^{N} AT_i^3 + b\sum_{i=1}^{N} AV_i^2 + c\sum_{i=1}^{N} DF_i^2}{N}, \quad \frac{\sum_{i=1}^{N} AT_i}{N}, \quad \frac{\sum_{i=1}^{N} AV_i}{N} \quad e \quad \frac{\sum_{i=1}^{N} DF_i}{N}$$

onde N representa o número de projectos por problema.

Conforme se pode observar o algoritmo *GaFolgaMod* é claramente superior aos outros dois algoritmos em todos os aspectos. Em termos absolutos é de salientar que o algoritmo *GaFolgaMod* obteve, para todas as instâncias, atrasos, avanços e desvios de fluxo muito próximos do valor óptimo (zero).

Os resultados detalhados relativos a cada um dos algoritmos são apresentados em Mendes (2003).

Nº projectos	Algoritmo	Mérito		Atrasos		Avanços		Desvio de fluxo	
		Média[1]	Desvio padrão[1]	Média[2]	Desvio padrão[2]	Média[3]	Desvio padrão[3]	Média[4]	Desvio padrão[4]
	GaFolgaMod	**15.55**	**29.04**	**0.00**	**0.00**	**1.17**	**1.44**	**0.22**	**0.43**
10	*GaFolgaNA*	20.74	21.08	0.00	0.00	2.25	1.35	0.29	0.35
	GaGen	996.57	1332.56	0.14	0.30	10.15	6.27	5.05	3.22
	GaFolgaMod	**4.26**	**10.99**	**0.00**	**0.00**	**0.59**	**0.35**	**0.11**	**0.30**
20	*GaFolgaNA*	42.94	124.61	0.00	0.01	1.47	1.67	0.33	0.74
	GaGen	707.30	653.73	0.09	0.11	7.46	5.21	0.49	0.46
	GaFolgaMod	**6.59**	**26.42**	**0.00**	**0.00**	**0.64**	**0.75**	**0.04**	**0.08**
30	*GaFolgaNA*	11.32	36.23	0.00	0.01	0.87	0.93	0.08	0.15
	GaGen	265.18	253.86	0.01	0.02	4.50	3.65	0.28	0.35
	GaFolgaMod	**0.56**	**0.23**	**0.00**	**0.00**	**0.49**	**0.05**	**0.01**	**0.04**
50	*GaFolgaNA*	1.36	2.24	0.00	0.00	0.54	0.12	0.03	0.02
	GaGen	268.25	354.95	0.01	0.03	2.91	2.09	0.08	0.10

Na **tab.6.2** apresentam-se os resultados obtidos pelo algoritmo *GaFolgaMod* com a função objectivo (6.5) e quando o termo $\sum_i DF_i^2$ desta é substituído por $\sum_i \frac{(DC_i - DI_i)^2}{DCC_i}$.

tab. 6.2 Resultados dos testes computacionais obtidos pelo algoritmo GaFolgaMod.

Nº projectos	Mérito		Atrasos		Avanços		Tempo de fluxo	
	1)	2)	1)	2)	1)	2)	1)	2)
10	15.55	53.45	0.00	0.00	1.17	1.11	100.15	99.12
20	4.26	84.11	0.00	0.00	0.59	0.53	95.37	95.28
30	6.59	42.07	0.00	0.00	0.64	0.56	94.90	94.98
50	0.56	38.94	0.00	0.00	0.49	0.49	94.79	94.79

1) média obtida usando $\sum_i DF_i^2$, 2) média obtida usando $\sum_i \frac{(DC_i - DI_i)^2}{DCC_i}$.

O algoritmo foi implementado em Visual Basic 6.0 e os testes computacionais foram realizados num computador com sistema operativo Windows Me e um processador AMD a 1.33 GHz. Os tempos médios de computação para 50 gerações por tipo de problema foram:

Número de Projectos	10	20	30	50
Tempo Médio para 50 Gerações	178 s	449 s	840 s	1860 s

6.6 Conclusões

Neste capítulo apresentou-se uma nova abordagem para o problema do planeamento de múltiplos projectos com recursos limitados. Esta nova abordagem combina uma inovadora medida de desempenho, um algoritmo genético e um esquema de geração de planos.

O algoritmo genético utiliza como alfabeto chaves aleatórias em vez da tradicional codificação binária. A programação das actividades é feito com recurso a prioridades e tempos de espera definidos pelo algoritmo genético. Os planos são obtidos através de um esquema de geração de planos activos parametrizados especificamente desenvolvido.

Os testes de desempenho do algoritmo foram realizados num conjunto de problemas com 10, 20, 30 e 50 projectos (1200, 2400, 3600 e 6000 actividades, respectivamente), tendo demonstrado que o algoritmo *GaFolgaMod* é claramente superior aos outros dois algoritmos (*GaGen* e *GaFolgaNA*) em todos os aspectos. Em termos absolutos o algoritmo *GaFolgaMod* obteve, para todas as instâncias, atrasos, avanços e desvios de fluxo muito próximos dos valores óptimos (zero).

7.1 Introdução

Apresenta-se neste capítulo uma nova abordagem para a resolução do problema do planeamento de operações em ambiente *Job Shop*.

No problema do JSP a medida de desempenho geralmente mais utilizada é a minimização do tempo necessário para a conclusão de todas as tarefas (ordens de fabrico), i.e., a minimização do tempo de conclusão, medida normalmente designada por *makespan*.

7.2 Formulação do problema JSP

O problema do planeamento de operações, designado por problema JSP, consiste na determinação de um plano para o processamento de n ordens de fabrico ("*job*"), em m máquinas que se encontram numa oficina ("*shop*"). Cada máquina processa uma operação de cada vez e após se ter iniciado o processamento de uma operação esta deve ser concluída sem interrupção. As operações de uma ordem de fabrico devem ser processadas de acordo com uma ordem pré-definida. O problema consiste em determinar um plano de fabrico das operações nas máquinas, considerando as restrições de precedência e capacidade, que minimize o *makespan* (C_{max}), i.e., o tempo de fim da última operação no plano.

Em termos de formulação, o problema do planeamento ou programação de operações em ambiente *Job Shop* pode ser descrito da seguinte forma: seja $J=\{0,1,\ldots,n,n+1\}$ o conjunto das operações a serem programadas e $M=\{1,\ldots,m\}$ o conjunto das máquinas. As operações 0 e $n+1$ são fictícias, têm duração nula e representam as operações inicial e final. As operações estão interrelacionadas por dois tipos de restrições:

- Restrições de precedência que implicam que cada operação j só pode ser programada após todas as suas operações precedentes, P_j, terem sido concluídas.
- Restrições de máquina que implicam que uma operação só pode ser programada se a máquina na qual se realiza a operação está disponível.

O problema do JSP é um caso particular do problema RCPSP abordado no capítulo 5. Cada máquina m pode ser considerada como um recurso cuja capacidade

cessora.

Considere-se que F_j representa o tempo de fim da operação j. Um plano pode ser representado por um vector de tempos de fim das operações $(F_1, , F_2, \ldots , F_{n+1})$. Seja $A(t)$ o conjunto das operações que estão em processamento no instante t, e seja $r_{j,m}=1$ se a operação j requer a máquina m para ser processada e $r_{j,m}=0$ caso contrário. Cada máquina tem capacidade disponível unitária e d_j representa a duração da operação j.

O modelo conceptual do problema JSP pode ser descrito da seguinte forma:

$$\text{Minimizar } F_{n+1} \ (C_{max}) \tag{7.1}$$

Sujeita a:

$$F_k \leq F_j - d_j \qquad j=1,\ldots,n+1 \ ; \ k \in P_j \tag{7.2}$$

$$\sum_{j \in A(t)} r_{j,m} \leq 1 \qquad m \in M \ ; \ t \geq 0 \tag{7.3}$$

$$F_j \geq 0 \qquad j=1,\ldots,n+1 \tag{7.4}$$

A função objectivo (7.1) minimiza o tempo de fim da operação $n+1$, portanto minimiza o *makespan*. As restrições (7.2) impõem as relações de precedência entre operações e as restrições (7.3) estabelecem que uma máquina só pode processar uma operação de cada vez. Finalmente (7.4) impõe que as variáveis de decisão sejam não negativas.

7.3 Nova abordagem

A nova abordagem apresentada neste capítulo combina um algoritmo genético com um procedimento que gera planos activos parametrizados e incorpora um procedimento de melhoria local. Em termos gerais a abordagem consiste nas seguintes três fases:

Construção de planos Esta fase recorre a um esquema de geração de planos do tipo paralelo modificado para construir planos activos parametrizados. A es-

colha da actividade a seleccionar em cada iteração do algoritmo é controlada através da prioridade e do tempo de espera associado a cada operação;

Procedimento de melhoria local Esta fase é usada para melhorar a solução obtida após a execução das fases anteriores;

Evolução das prioridades e dos tempos de espera Esta fase utiliza um algoritmo genético para evoluir (melhorar) as prioridades e tempos de espera utilizadas na fase anterior.

Nas secções seguintes serão apresentados os detalhes da abordagem.

7.3.1 Esquema de geração de planos activos parametrizados

O esquema de geração de planos utilizado para construir planos activos parametrizados combina a lógica dos esquemas série e paralelo descritos no capítulo 2.

Com vista a poder definir formalmente o algoritmo apresenta-se seguidamente alguma notação. Para cada iteração g, existe um tempo t_g. O conjunto activo de actividades compreende todas as operações que estão activas em t_g, i.e. $A_g = \{ j \in J \mid F_j - d_j \leq t_g < F_j \}$. A capacidade disponível da máquina m no instante t_g é dada por . S_g é o conjunto de todas as operações programadas até à iteração g, e T_g representa os tempos de conclusão das operações em S_g. Seja $tEspera_g$ o tempo de espera associado com a iteração g, e E_g o conjunto de todas as operações que podem ser programadas (cujas predecessoras já foram programadas) no intervalo $[t_g, t_g + tEspera_g]$, i.e.,

$$E_g = \{ j \in J \setminus S_{g-1} \mid F_i \leq t_g + tEspera_g \, (i \in P_j) \}$$

A **fig. 7.1** descreve o pseudo-código do procedimento utilizado para gerar planos activos parametrizados.

fig. 7.1 Pseudo-código para gerar planos activos parametrizados.

Inicialização: $g = 1, t_1 = 0, A_0 = \{0\}, \mathcal{T}_0 = \{0\}, S_0 = \{0\}, RD_m(0) = 1\ (m \in M)$

Enquanto $|S_g| < n + 2$ **Repetir**

{

Actualizar Eg

Enquanto $E_g \neq \{\}$ **Repetir**

{

Seleccionar operação com mais prioridade

$$j^* = \underset{j \in E_g}{\operatorname{argmax}} \left\{ PRIORIDADE_j \right\}$$

Calcular tempo de fim mais cedo (apenas em termos de precedência)

$$FMC_{j^*} = \max_{i \in P_{j^*}} \{F_i\} + d_{j^*}$$

Calcular tempo de fim mais cedo (em termos de precedência e capacidade)

$$F_{j^*} = \min \{\ t \in \left[FMC_{j^*} - d_{j^*}, \infty \right] \cap F_g \,|\, r_{j^*,m} \leq RD_m(\tau),$$

$$r_{j^*,m} > 0, \tau \in \left[t, t + d_{j^*} \right] \} + d_{j^*}$$

Actualizar $S_g = S_{g-1} \cup \left\{ j^* \right\}, \mathcal{T}_g = \mathcal{T}_{g-1} \cup \left\{ F_{j^*} \right\}$

Incrementar iteração $g=g+1$

Actualizar $A_g, E_g, RD_m(t) \,|\, t \in \left[F_{j^*} - d_{j^*}, F_{j^*} \right], m \in M | r_{j^*,m} > 0$

}

Determinar o tempo associado à iteração g

$t_g = \min \left\{ t \in T_{g-1} \,|\, t > t_{g-1} \right\}$

}

7.3.2 Algoritmo genético

Seguidamente descrevem-se os detalhes do algoritmo genético utilizado.

.3.2.1 Representação cromossómica

Cada cromossoma (solução) é composto por $2n$ genes onde n representa o número de operações. Os primeiros n genes são usados para obter as prioridades associadas a cada operação e os últimos n genes são usados para obter os tempos de espera associados a cada iteração, conforme ilustra o esquema seguinte.

$$\text{Cromossoma} = (\underbrace{\text{gene}_1, \ldots, \text{gene}_n}_{\text{Prioridades}}, \underbrace{\text{gene}_{n+1}, \ldots, \text{gene}_{2n}}_{\text{Tempos de espera}})$$

.3.2.2 Descodificação das prioridades das operações

Conforme mencionado anteriormente os primeiros n genes são associados a cada operação. Para descodificação das prioridades associadas a cada operação foi utilizada a seguinte expressão:

$$PRIORIDADE_j = Gene_j.$$

.3.2.3 Descodificação dos tempos de espera

Conforme mencionado anteriormente os genes de $n+1$ até $2n$ são usados para obter os tempos de espera associados a cada iteração. O tempo de espera, $tEspera_g$, usado para realizar a programação na iteração g, é determinado pela seguinte expressão:

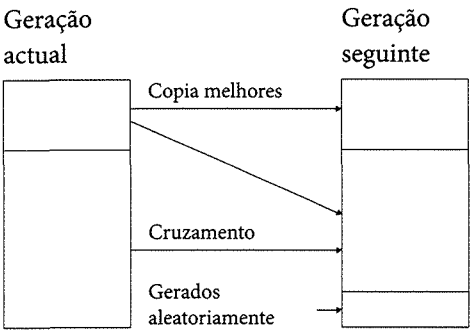
$$tEspera_g = Gene_{n+g} \times 1.5 \times MaxTempoProc \qquad g = 1, \ldots, n$$

na qual *MaxTempoProc* é o maior tempo de processamento de entre todas as operações do projecto.

.3.2.4 Estratégia evolutiva

A estratégia evolutiva segue o esquema descrito na secção 4.5.1, cujo esquema geral reproduzimos na **fig. 7.2**, existindo algumas alterações a nível da geração da população inicial e do processo de mutação/imigração.

fig. 7.2 Transição de geração.



Com vista a garantir a diversidade de planos incluídos nas populações tanto no processo de inicialização (primeira geração) como no processo de mutação/imigração usado para criar novas populações, impôs-se que 25% dos planos da população inicial e resultantes do processo de mutação/imigração fossem do tipo não-atrasado, i.e, com tempos de espera associados a todas as iterações iguais a zero, ou seja,

$$tEspera_g = 0 \qquad g = 1, \ldots, n.$$

O valor da percentagem de planos não atrasados, igual a 25%, a incluir na população inicial e resultante do processo de mutação/imigração foi obtido por testes experimentais, sendo este o valor que obteve melhor desempenho de entre os valores 0% a 100% com intervalos de 5%.

7.3.3 Procedimento de melhoria local

O esquema de geração de planos activos parametrizados não garante a construção de planos localmente óptimos, pelo que nalguns casos é possível melhorar a solução obtida através de procedimentos de melhoria local. Neste trabalho utiliza-se um procedimento de melhoria local baseado na vizinhança de Nowicki e Smutnicki (1996).

O procedimento de melhoria local tem início após a obtenção de um plano (solução corrente), ver **fig. 7.3**, e começa com a identificação do caminho crítico da solução obtida, ver **fig. 7.4**. Designar-se-á por operação crítica uma operação que pertence ao caminho crítico. É possível decompor o caminho crítico num conjunto de blocos, onde um bloco é constituído por um número máximo de operações críticas adjacentes, que requerem a mesma máquina, ver **fig. 7.4**.

fig. 7.3 Exemplo de solução corrente.

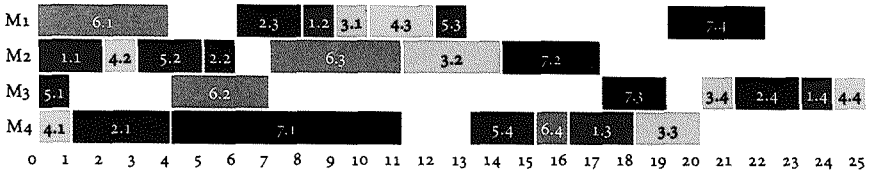
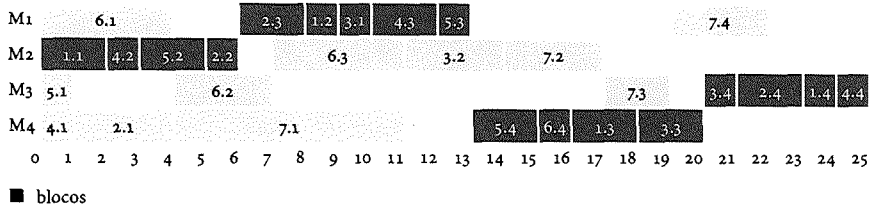
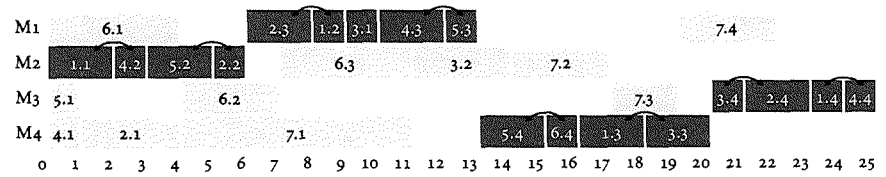


fig. 7.4 Exemplo do caminho crítico da solução corrente.



A abordagem utilizada para melhoria local, conforme já mencionado, baseia-se na vizinhança proposta por Nowicki e Smutnicki (1996). Sendo dados b blocos críticos, se $1 < l < b$, então trocar apenas as duas últimas e as duas primeiras operações do bloco. Caso contrário, se $l = 1$ (b) trocar apenas as últimas (primeiras) operações do bloco, ver **fig. 7.5**. No caso do primeiro ou último bloco conterem apenas duas operações estas devem ser trocadas. Se o bloco contém apenas uma operação, então não há qualquer troca a realizar.

fig. 7.5 Trocas possíveis de acordo com vizinhança de Nowicki e Smutnicki (1996).



Apresenta-se na **fig. 7.6** um exemplo da aplicação da vizinhança de Nowicki e Smutnicki (1996). A solução corrente tem um *makespan* de 25 unidades de tempo. Ir-se-á proceder à troca de sequência entre as operações 4.3 e 5.3 pertencentes ao bloco crítico da máquina M1. Após a troca de sequência das operações 4.3 e 5.3 obtém-se um novo plano cujo *makespan* é passa para 24 unidades de tempo, ver **fig. 7.7**.

fig. 7.6 Plano antes da troca de sequência entre as operações 4.3 e 5.3.

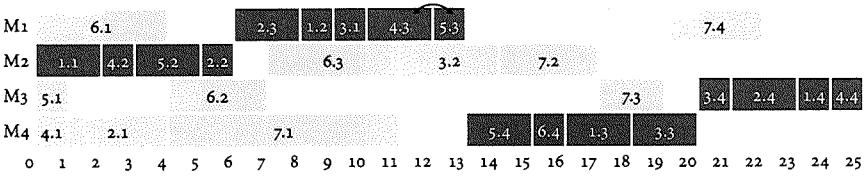
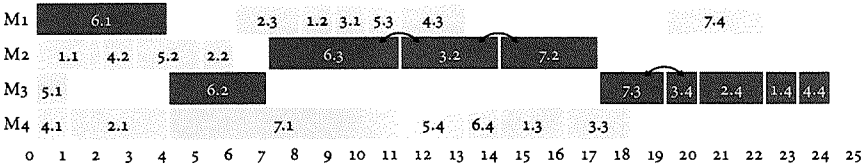


fig. 7.7 Plano após troca de sequência entre as operações 4.3 e 5.3.



O procedimento de melhoria local poderia continuar considerando o plano apresentado na **fig. 7.7** como plano corrente. De notar que neste caso de acordo com a vizinhança de Nowicki e Smutnicki (1996) existem apenas três trocas possíveis 6.3-3.2, 3.2-7.2 e 7.3-3.4, conforme ilustrado pelos arcos na **fig. 7.7**.

O algoritmo de melhoria local consiste na aplicação sucessiva de trocas entre operações que satisfazem as condições de vizinhança de Nowicki e Smutnicki (1996). Uma troca é aceite se o valor do *makespan* for melhorado. Caso contrário, a troca não é aceite e permanece o plano obtido antes da troca. Uma vez realizada uma troca, é necessário identificar o novo caminho crítico. O processo de melhoria local termina quando nenhuma das trocas possíveis melhora o valor do *makespan*. O pseudo-código relativo ao algoritmo de melhoria local é apresentado na **fig. 7.8**.

fig. 7.8 Pseudo-código do algoritmo de melhoria local.

```
Melhoria_Local (SoluçãoCorrente)

SoluçãoCorrenteActualizada = Verdadeiro
Enquanto SoluçãoCorrenteActualizada Repetir
{
  SoluçãoCorrenteActualizada = Falso
  Determinar o caminho crítico e todos os blocos críticos da SoluçãoCorrente
  Enquanto Houver blocos não processados e Não SoluçãoCorrenteActualizada Repetir
  {
    Se Não Primeiro Bloco Crítico Então
    {
      NovaSolução = Troca duas primeiras operações do bloco em SoluçãoCorrente
      Se Makespan (NovaSolução) < Makespan (SoluçãoCorrente) Então
      {
        SoluçãoCorrente = NovaSolução
        SoluçãoCorrenteActualizada = Verdadeiro
      }
    }

    Se Não Último Bloco Crítico e Não SoluçãoCorrenteActualizada Então
    {
      NovaSolução = Troca duas últimas operações do bloco em SoluçãoCorrente
      Se Makespan (NovaSolução) < Makespan (SoluçãoCorrente) Então
      {
        SoluçãoCorrente = NovaSolução
        SoluçãoCorrenteActualizada = Verdadeiro
      }
    }
  }
}
```

7.4 Testes Experimentais

Esta secção apresenta os resultados dos testes experimentais obtidos pelo algoritmo HGA-JSP (*Hybrid Genetic Algorithm for JSP*).

Os testes experimentais foram realizados sobre dois conjuntos de problemas. O primeiro conjunto de problemas padrão consiste em 3 problemas propostos por Fischer e Thompson (1963), um de 6 ordens de fabrico em 6 máquinas (FT06), outro de 10 ordens de fabrico em 10 máquinas (FT10) e outro de 20 ordens de fabrico em 5 máquinas (FT20). O segundo conjunto de problemas consiste em 40 problemas propostos por Lawrence (1984), de LA01 a LA40, com tamanhos variando entre 10 a 30 ordens de fabrico e 5 a 15 máquinas. Estes problemas são normalmente os mais utilizados nos testes de desempenho para algoritmos de optimização de programação de operações.

Para a realização dos testes experimentais o algoritmo genético usou a seguinte configuração para todos os problemas:

Tamanho da população	2 × número de actividades do problema
Cruzamento	probabilidade de cruzamento = 0.7
Selecção	10% dos melhores cromossomas são copiados para a próxima geração
Mutação	20% dos cromossomas da população
Mérito	*makespan* (a minimizar)
Sementes	20
Critério de paragem	400 Gerações.

O algoritmo proposto é comparado com os seguintes algoritmos:

Problem And Heuristic Space
Storer et al. (1992)

Genetic Algorithms
Aarts et al. (1994)
Croce et al (1995)
Dorndorf e Pesch (1995)
Gonçalves e Beirão (1999)

GRASP
Binato et al. (2002)
Aiex et al. (2003)

Hybrid Genetic e Simulate Annealing
Wang and Zheng (2001)
Tabu Search
Nowicki and Smutnicki (1996).

O algoritmo foi implementado em Visual Basic 6.0 e os testes foram realizados num computador com um AMD Thunderbird CPU a 1.333 GHz e sistema operativo MS Windows Me. A **tab. 7.1** apresenta os resultados listando o nome do problema, a dimensão do problema, (numero de ordens de fabrico × numero de operações), a melhor solução conhecida (BKS), o tempo de CPU (em segundos) para 400 gerações do algoritmo genético e a solução para cada um dos algoritmos.

A **tab. 7.2** mostra o número de instancias resolvidas (NIR), e a média dos desvios relativos, com respeito aos valores de BKS (MDR). O MDR foi calculado para o Hybrid Genetic Algorithm (HGA-JSP) e para os outros algoritmos (OA). A última coluna (Melhoria), apresenta a redução em MDR obtido pelo algoritmo HGA-JSP com respeito a cada um outros algoritmos.

tab. 7.1 Resultados experimentais.

Instancia	Tam.	BKS	HGA-JSP	Time (sec.)	Wang e Zheng (2001)	Aiex et al. (2003)	Binato et al. (2002)	Nowicki e Smutnicki (1996)	Gonçalves e Beirão (1999)	Croce et al.. (1995)	Dorndorf & Pesch			Aarts et al.		Storer et al. (1992)
											P-GA (1995)	SBGA (40) (1995)	SBGA (60) (1995)	GLS1 (1994)	GLS2 (1994)	
FT06	6x6	55	55	13	55	55	55	55	55		–					
FT10	10x10	930	930	292	930	930	938	930	936	946	960			935	945	952
FT20	20x5	1165	1165	204	1165	1165	1169	1165	1177	1178	1249			1165	1167	
LA01	10x5	666	666	37	666	666	666	666	666	666	666	666		666	666	666
LA02	10x5	655	655	51		655	655	655	666	666	681	666		668	659	
LA03	10x5	597	597	39		597	604	597	597	666	620	604		613	609	
LA04	10x5	590	590	42		590	590	590	590	–	620	590		599	594	
LA05	10x5	593	593	32		593	593	593	593	–	593	593		593	593	
LA06	15x5	926	926	99	926	926	926	926	926	926	926	926		926	926	
LA07	15x5	890	890	86		890	890	890	890	–	890	890		890	890	
LA08	15x5	863	863	99		863	863	863	863	–	863	863		863	863	
LA09	15x5	951	951	94		951	951	951	951	–	951	951		951	951	
LA10	15x5	958	958	91		958	958	958	958	–	958	958		958	958	
LA11	20x5	1222	1222	197	1222	1222	1222	1222	1222	1222	1222	1222		1222	1222	
LA12	20x5	1039	1039	201		1039	1039	1039	1039	–	1039	1039		1039	1039	
LA13	20x5	1150	1150	189		1150	1150	1150	1150	–	1150	1150		1150	1150	
LA14	20x5	1292	1292	187		1292	1292	1292	1292	–	1292	1292		1292	1292	
LA15	20x5	1207	1207	187		1207	1207	1207	1207	–	1237	1207		1207	1207	
LA16	10x10	945	945	232	945	945	946	945	977	979	1008	961	961	977	977	981

LA17	10X10	784	784	216		784	784	784	787	–	809	787	784	791	791	794
LA18	10X10	848	848	219		848	848	848	848	–	916	848	848	856	858	860
LA19	10X10	842	842	235		842	842	842	857	–	880	863	848	863	859	860
LA20	10X10	902	907	235		902	907	902	910	–	928	911	910	913	916	
LA21	15X10	1046	1046	602	1058	1057	1091	1047	1047	1097	1139	1074	1074	1084	1085	
LA22	15X10	927	935	629		927	960	927	936	–	998	935	936	954	944	
LA23	15X10	1032	1032	594		1032	1032	1032	1032	–	1072	1032	1032	1032	1032	
LA24	15X10	935	953	578		954	978	939	955	–	1014	960	957	970	981	
LA25	15X10	977	986	609		984	1028	977	1004	–	1014	1008	1007	1016	1010	
LA26	20X10	1218	1218	1 388	1218	1218	1271	1218	1218	1231	1278	1219	1218	1240	1236	
LA27	20X10	1235	1256	1 251		1269	1320	1236	1260	–	1378	1272	1269	1308	1300	
LA28	20X10	1216	1232	1 267		1225	1293	1216	1241	–	1327	1240	1241	1281	1265	
LA29	20X10	1157	1196	1 350		1203	1293	1160	1190	–	1336	1204	1210	1290	1260	
LA30	20X10	1355	1355	1 260		1355	1368	1355	1356	–	1411	1355	1355	1402	1386	
LA31	30X10	1784	1784	3 745	1784	1784	1784	1784	1784	1784	–			1784	1784	
LA32	30X10	1850	1850	3 741		1850	1850	1850	1850	–	–			1850	1850	
LA33	30X10	1719	1719	3 637		1719	1719	1719	1719	–	–			1719	1719	
LA34	30X10	1721	1721	3 615		1721	1753	1721	1730	–	–			1737	1730	
LA35	30X10	1888	1888	3 716		1888	1888	1888	1888	–	–			1894	1890	
LA36	15X15	1268	1279	1 826	1292	1287	1334	1268	1305	1305	1373	1317	1317	1324	1311	1305
LA37	15X15	1397	1408	1 860		1410	1457	1407	1441	–	1498	1484	1446	1449	1450	1458
LA38	15X15	1196	1219	1 859		1218	1267	1196	1248	–	1296	1251	1241	1285	1283	1239
LA39	15X15	1233	1246	1 869		1248	1290	1233	1264	–	1351	1282	1277	1279	1279	1258
LA40	15X15	1222	1241	2 185		1244	1259	1229	1252	–	1321	1274	1252	1273	1260	1258

tab. 7.2 Média dos desvios relativos.

Algoritmo	NIR	MDR		Melhoria de HGA-JSP
		OA	HGA-JSP	
Problem and Heuristic Space				
Storer et al. (1992)	11	2.44%	0.56%	1.88 %
Genetic Algorithms				
Aarts et al. (1994) - GLS1	42	1.97%	0.40%	1.57 %
Aarts et al. (1994) - GLS2	42	1.71%	0.40%	1.31 %
Croce et al (1995)	12	2.37%	0.07%	2.30 %
Dorndorf et al. (1995) - PGA	37	4.61%	0.46%	4.15 %
Dorndorf et al. (1995) - SBGA (40)	35	1.42%	0.48%	0.94 %
Dorndorf et al. (1995) - SBGA (60)	20	1.94%	0.84%	1.10 %
Gonçalves e Beirão (1999)	43	0.90%	0.39%	0.51 %
GRASP				
Binato et al. (2002)	43	1.77%	0.39%	1.38 %
Aiex et al. (2001)	43	0.43%	0.39%	0.04 %
Hybrid Genetic e Simulated Annealing				
Wang e Zheng (2001)	11	0.28%	0.08%	0.20 %
Tabu Search				
Nowicki e Smutnicki (1996)	43	0.05 %	0.39%	-0.34 %

O algoritmo HGA-JSP obteve um valor de MDR = 0.39% tendo alcançado a solução óptima em 31 das instâncias (72%). O algoritmo HGA-JSP apresenta melhorias em relação a todos os outros algoritmos com excepção do algoritmo de Nowicki e Smutnicki, que apresenta melhores valores nos problemas de 15 x 15.

7.5 Conclusões

Neste capítulo apresenta-se uma nova abordagem para o problema do planeamento de operações em ambiente *Job Shop*. Esta abordagem combina um algoritmo genético, um esquema de geração de planos e um procedimento de melhoria local.

O algoritmo genético utiliza como alfabeto chaves aleatórias em vez da tradicional codificação binária. O planeamento das operações é realizado com recurso a prioridades e tempos de espera definidos pelo algoritmo genético. Os planos são obtidos através de um esquema de geração de planos activos parametrizados especificamente desenvolvido. Para além da combinação de um algoritmo gené-

tico com um procedimento de geração de planos activos parametrizados, incorpora-se também um procedimento de melhoria local baseado na vizinhança de Nowicki e Smuttnicki (1996).

Os testes de desempenho do algoritmo, realizados com base em 3 problemas propostos por Fischer e Thompson (1963), um de 6 ordens de fabrico em 6 máquinas (FT06), outro de 10 ordens de fabrico em 10 máquinas (FT10) e outro de 20 ordens de fabrico em 5 máquinas (FT20) e em 40 problemas propostos por Lawrence (1984), de LA01 a LA40, com tamanhos variando entre 10 a 30 ordens de fabrico e 5 a 15 máquinas, demonstram a boa qualidade das soluções obtidas pela heurística HGA-JSP.

08

CONSIDERAÇÕES FINAIS

Esta publicação apresenta uma nova abordagem para resolver o problema do planeamento de projectos com recursos limitados. Devido à sua natureza combinatória o problema do planeamento de projectos com recursos limitados é considerado um problema de elevada complexidade.

Os geradores de planos descritos combinam algoritmos genéticos com um novo procedimento capaz de gerar uma nova classe de planos - planos activos parametrizados.

O sistema desenvolvido permite obter soluções para os seguintes tipos de problemas:

- Planeamento de projectos com recursos limitados – RCPSP;
- Planeamento de múltiplos projectos com recursos limitados – RCMPSP;
- Planeamento de operações em ambientes do tipo *Job Shop* – JSP.

No caso particular do planeamento de operações em ambientes do tipo *Job Shop*, para além da combinação de um algoritmo genético com um procedimento de geração de planos activos parametrizados, incorpora-se também um procedimento de melhoria local baseado na vizinhança de Nowicki e Smuttnicki (1996).

Com vista a demonstrar a eficiência e eficácia dos algoritmos propostos foram realizados vários testes experimentais. No caso dos problemas RCPSP e JSP. foram realizados testes com base em conjuntos de problemas padrão conhecidos da literatura. Para o problema RCMPSP foi desenvolvido um gerador de problemas específico capaz de gerar problemas com solução óptima conhecida.

Para o problema do RCPSP, os testes de desempenho dos algoritmos foram realizados com base nos problemas dos conjuntos J30, J60 e J120 apresentados em Kolisch et al. (1998) e disponíveis na biblioteca PSPLIB. O algoritmo GAPS-III obtém, em termos relativos, um desempenho superior quando comparado com as heurísticas apresentadas em Hartmann e Kolisch (2000). Adicionalmente, para os problemas dos conjuntos J30, J60 e J120, o algoritmo GAPS-III obtêm, em termos absolutos, também um desempenho superior quando comparado com os melhores valores conhecidos para os problemas dos conjuntos J30, J60 e J120, de acordo com Hartmann e Kolisch (2000).

Para o problema do RCPSP, os testes de desempenho do algoritmo desenvolvido foram realizados num conjunto de problemas com 10, 20, 30 e 50 projectos (1200, 2400, 3600 e 6000 actividades, respectivamente). O algoritmo *GaFolgaMod* é claramente superior aos outros dois algoritmos (*GaGen e GaFolgaNA*) em todos os

aspectos. Em termos absolutos o algoritmo *GaFolgaMod* obteve, para todas as instâncias, atrasos, avanços e desvios de fluxo muito próximos dos valores óptimos (zero).

Para o problema do JSP os testes de desempenho do algoritmo, realizados com base em 43 problemas padrão conhecidos da literatura, demonstram a boa qualidade das soluções obtidas pelo algoritmo HGA-JSP, sendo de salientar a obtenção do valor óptimo para o desafiante problema de 10x10 de Fischer e Thompson (1963).

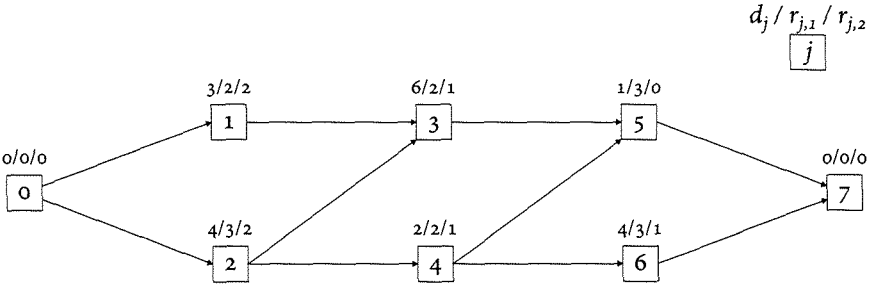
Os resultados dos testes confirmam a excelente qualidade dos geradores de soluções desenvolvidos bem como a capacidade dos algoritmos genéticos para abordar estes tipos de problemas.

Podemos ainda concluir que a construção de planos activos parametrizados adiciona aos algoritmos uma elevada eficiência e eficácia. Um outro factor que confere maior eficácia aos algoritmos é a inclusão de conhecimento específico do problema.

Rede de projecto

Na **fig. A.1** ilustra-se uma rede de projecto do tipo AON. As actividades são representadas por j, as durações destas por d_j e a utilização do recurso i pela actividade j por $r_{j,i}$. Neste exemplo as actividades inicial e final de projecto são respectivamente a actividade 0 e a actividade 7, existem 8 actividades (0, 1, 2, 3, 4, 5, 6, 7) e dois recursos (R1, R2) cujas capacidades são respectivamente 4 e 2.

fig. A.1 Exemplo de rede de projecto com representação AON.



Abordagem GA-SLACKMOD

Determinação de plano activo parametrizado

A **tab. A.1** apresenta os valores das prioridades, das actividades, e dos tempos de espera, de cada iteração, a serem utilizados no exemplo que ilustra o esquema de geração de planos activos parametrizados.

tab. A.1 Prioridades e tempos de espera usados no exemplo.

Actividade / Iteração j/g	$PRIORIDADE_j$	$tEspera_g$
0	0.38	0.00
1	0.37	1.98
2	0.40	2.31
3	0.47	3.96
4	0.51	4.13
5	0.72	7.25
6	0.34	8.05
7	0.91	0.00

- **Inicialização / Iteração $g=0$**

$t_1 = 0$, $A_0 = \{0\}$, $\mathcal{F}_0 = \{0\}$, $S_0 = \{0\}$,

$RD_k(t)=$

		t															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	>15
k	1	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

- **Iteração $g=1$**

Calcular $E_1 = \{ j \in J \setminus S_0 \mid F_i \leq 0 + 1.98(i \in P_j) \} = \{1, 2\}$

Seleccionar actividade com maior prioridade

$$j^* = \underset{j \in E_1}{\operatorname{argmax}} \{ 0.37 , 0.40 \} = 2$$

Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

$FMC_2 = 0 + 4 = 4$

Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

$F_2 = 0 + 4 = 4$

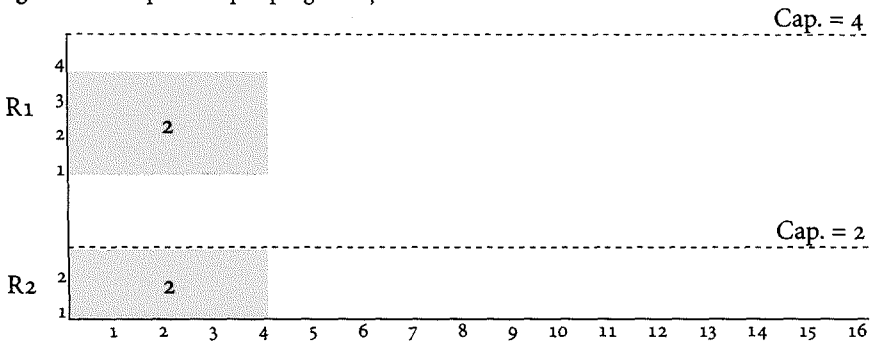
$S_1 = S_0 \cup \{2\} = \{0\} \cup \{2\} = \{0, 2\}$

$T_1 = T_0 \cup \{4\} = \{0\} \cup \{4\} = \{0, 4\}$

$g = 1 + 1 = 2$

O plano parcial do esquema de geração de planos activos parametrizados após a iteração $g=1$ pode ser visualizado na **fig.A.2.**

fig. A.2 Plano parcial após programação da actividade 2.



- **Iteração $g=2$**

Actualizar $A_2 = \{0,2\}$

Actualizar $E_2 = \{ j \in J \setminus S_1 \mid F_i \leq 0 + 2.31\,(i \in P_j) \} = \{1\}$

$RD_k(t)=$

k \ t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	>15
1	1	1	1	1	4	4	4	4	4	4	4	4	4	4	4	4
2	0	0	0	1	2	2	2	2	2	2	2	2	2	2	2	2

Seleccionar actividade com maior prioridade

$$j^* = \underset{j \in E_2}{\operatorname{argmax}} \{0.37\} = 1$$

Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

$FMC_1 = 0 + 3 = 3$

Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

$F_1 = 4 + 3 = 7$

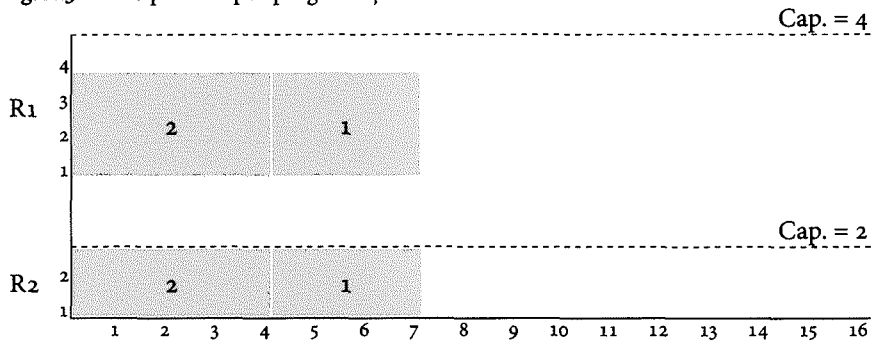
$S_2 = S_1 \cup \{1\} = \{0,2\} \cup \{1\} = \{0,1,2\}$

$F_2 = F_1 \cup \{7\} = \{0,4\} \cup \{7\} = \{0,4,7\}$

$g = 2 + 1 = 3$

O plano parcial do esquema de geração de planos activos parametrizados após a iteração $g=2$ pode ser visualizado na **fig. A.3**:

fig. A.3 Plano parcial após programação da actividade 1.



- **Iteração $g=3$**

Actualizar $A_3 = \{2\}$

Actualizar $E_3 = \{ j \in J \setminus S_2 \mid F_i \leq 0 + 3.96\ (i \in P_j) \} = \{\ \}$

$RD_k(t)=$

k \ t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	>15
1	1	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4
2	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2

Determinar o tempo associado à iteração $g=3$

$t_3 = \min\{4, 7\} = 4$

Actualizar $E_3 = \{ j \in J \setminus S_2 \mid F_i \leq 4 + 3.96\ (i \in P_j) \} = \{3, 4\}$

Seleccionar actividade com maior prioridade

$$j^* = \underset{j \in E_3}{\operatorname{argmax}} \{0.47, 0.51\} = 4$$

Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

$FMC_4 = 4 + 2 = 6$

Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

$F_4 = 7 + 2 = 9$

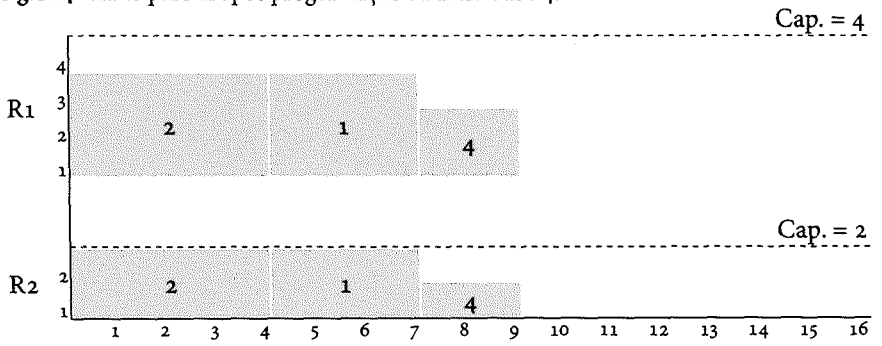
$S_3 = S_2 \cup \{4\} = \{0, 1, 2\} \cup \{4\} = \{0, 1, 2, 4\}$

$T_3 = T_2 \cup \{6\} = \{0, 4, 7\} \cup \{9\} = \{0, 4, 7, 9\}$

$g = 3 + 1 = 4$

O plano parcial do esquema de geração de planos activos parametrizados após a iteração $g = 3$ pode ser visualizado na **fig. A.4**:

fig. A.4 Plano parcial após programação da actividade 4.



- **Iteração $g=4$**

Actualizar $A_4=\{1\}$

Actualizar $E_4=\{j\in J\setminus S_3 \mid F_i\leq 4+4.13\,(i\in P_j)\}=\{3\}$

$RD_k(t)=$

k \ t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	>15
1	1	1	1	1	1	1	1	2	2	4	4	4	4	4	4	4
2	0	0	0	0	0	0	0	1	1	2	2	2	2	2	2	2

Seleccionar actividade com maior prioridade

$$j^*=\underset{j\in E_4}{\operatorname{argmax}}\{0.47\}=3$$

Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

$FMC_3=4+6=10$

Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

$F_3=7+6=13$

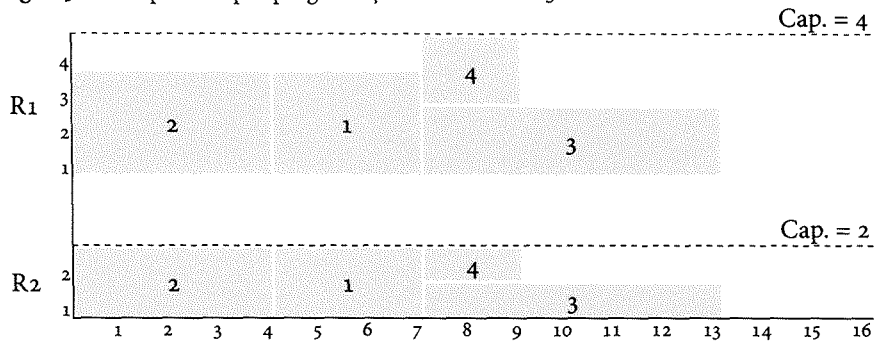
$S_4=S_3\cup\{3\}=\{0,1,2,4\}\cup\{3\}=\{0,1,2,3,4\}$

$T_4=T_3\cup\{13\}=\{0,4,7,9\}\cup\{13\}=\{0,4,7,9,13\}$

$\boldsymbol{g=4+1=5}$

O plano parcial do esquema de geração de planos activos parametrizados após a iteração $g=4$ pode ser visualizado na **fig. A.5**:

fig. A.5 Plano parcial após programação da actividade 3.



- **Iteração $g=5$**

Actualizar $A_5=\{1\}$

Actualizar $E_5=\{j\in J\setminus S_4 \mid F_i\leq 4+8.05\,(i\in P_j)\}=\{\ \}$

$RD_k(t)=$

k \ t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	>15
1	1	1	1	1	1	1	1	0	0	2	2	2	2	4	4	4
2	0	0	0	0	0	0	0	0	0	1	1	1	1	2	2	2

Determinar o tempo associado à iteração $g=5$

$t_5=\min\{7,9,13\}=7$

Actualizar $E_5=\{j\in J\setminus S_4 \mid F_i\leq 7+8.05\,(i\in P_j)\}=\{5,6\}$

Seleccionar actividade com maior prioridade

$j^*=\underset{j\in E_5}{\operatorname{argmax}}\{0.34,0.72\}=5$

Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

$FMC_5=13+1=14$

Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

$F_6 = 13 + 1 = 14$

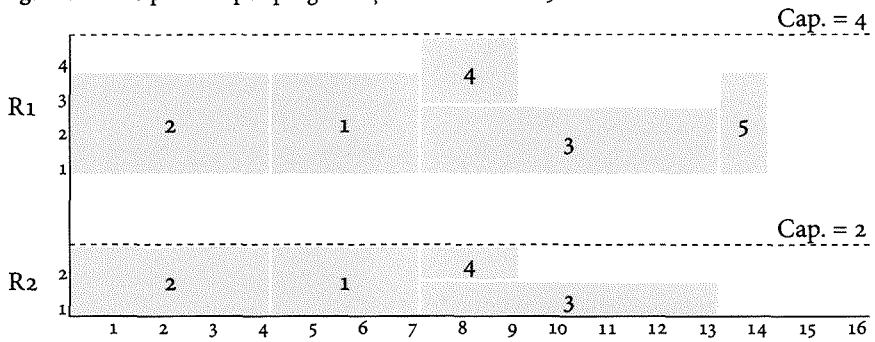
$S_5 = S_4 \cup \{5\} = \{0,1,2,3,4\} \cup \{5\} = \{0,1,2,3,4,5\}$

$T_5 = T_4 \cup \{14\} = \{0,4,7,9,13\} \cup \{14\} = \{0,4,7,9,13,14\}$

$\boldsymbol{g = 5 + 1 = 6}$

O plano parcial do esquema de geração de planos activos parametrizados após a iteração $g = 5$ pode ser visualizado na **fig. A.6**:

fig. A.6 Plano parcial após programação da actividade 5.



- **Iteração $g = 6$**

Actualizar $A_6 = \{3,4\}$

Actualizar $E_6 = \{ j \in J \setminus S_5 \mid F_i \leq 7 + 7.25\ (i \in P_j) \} = \{6\}$

$RD_k(t)$=

		t															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	>15
k	1	1	1	1	1	1	1	1	0	0	2	2	2	2	1	4	4
	2	0	0	0	0	0	0	0	0	0	1	1	1	1	2	2	2

Seleccionar actividade com maior prioridade

$$j^* = \underset{j \in E_6}{\operatorname{argmax}} \{0.34\} = 6$$

Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

$$FMC_6 = 14 + 4 = 18$$

Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

$$F_6 = 14 + 4 = 18$$

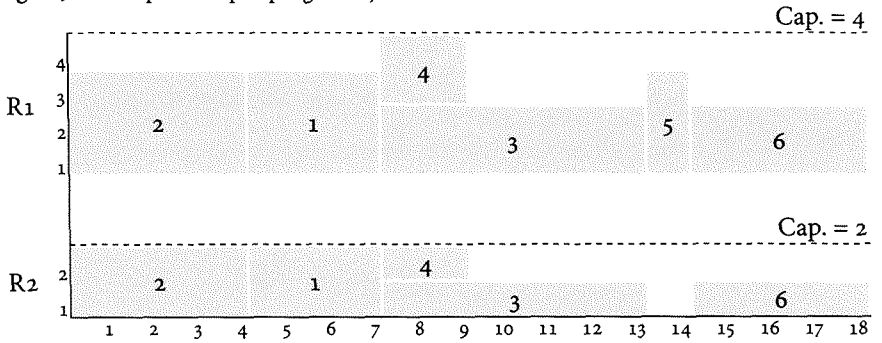
$$S_6 = S_5 \cup \{6\} = \{0,1,2,3,4,5\} \cup \{6\} = \{0,1,2,3,4,5,6\}$$

$$T_6 = T_5 \cup \{18\} = \{0,4,7,9,13,14\} \cup \{18\} = \{0,4,7,9,13,14,18\}$$

$$g = 6 + 1 = 7$$

O plano parcial do esquema de geração de planos activos parametrizados após a iteração $g = 6$ pode ser visualizado na **fig. A.7**:

fig. A.7 Plano parcial após programação da actividade 6.



- **Iteração $g = 7$**

  Actualizar $A_7 = \{3,4\}$

  Actualizar $E_7 = \{ j \in J \setminus S_6 \mid F_i \leq 7 + 0 \; (i \in P_j) \} = \{\}$

$RD_k(t)=$

k	t															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15-18	>18
1	1	1	1	1	1	1	1	0	0	2	2	2	2	1	2	4
2	0	0	0	0	0	0	0	0	0	1	1	1	1	2	1	2

Determinar o tempo associado à iteração $g=7$ até $E_g \neq \{\}$

$$t_7 = \min\{9,13,14,18\} = 9 \quad , \quad E_7 = \{ j \in J \setminus S_6 \mid F_i \leq 9 + 0 \; (i \in P_j) \} = \{\}$$

$$t_7 = \min\{13,14,18\} = 13 \quad , \quad E_7 = \{ j \in J \setminus S_6 \mid F_i \leq 13 + 0 \; (i \in P_j) \} = \{\}$$

$$t_7 = \min\{14,18\} = 14 \quad , \quad E_7 = \{ j \in J \setminus S_6 \mid F_i \leq 14 + 0 \; (i \in P_j) \} = \{\}$$

$$t_7 = \min\{18\} = 18 \quad , \quad E_7 = \{ j \in J \setminus S_6 \mid F_i \leq 18 + 0 \; (i \in P_j) \} = \{7\}$$

Seleccionar actividade com maior prioridade

$$j^* = \underset{j \in E_7}{\operatorname{argmax}} \{0.00\} = 7$$

Calcular tempo de fim mais cedo (tendo em conta apenas as restrições de precedência)

$$FMC_7 = 18 + 0 = 10$$

Calcular tempo de fim mais cedo (tendo em conta as restrições de precedência e de capacidade)

$$F_7 = 18 + 0 = 18$$

$$S_7 = S_6 \cup \{7\} = \{0,1,2,3,4,5,6\} \cup \{7\} = \{0,1,2,3,4,5,6,7\}$$

$$T_7 = T_6 \cup \{15\} = \{0,4,7,9,13,14,15,18\} \cup \{18\} = \{0,4,7,9,13,14,15,18\}$$

$$\boldsymbol{g = 7 + 1 = 8}$$

Após esta iteração o plano final está concluído pois a condição $|S_g| < n + 2$ é verdadeira. Dado que a actividade $7\,(n+1)$ não tem duração nem ocupa recursos, o plano final é igual ao plano representado na **fig. A.7.**

Algoritmo genético

Representação cromossómica

Cada cromossoma é composto por 2*8+1 genes onde n=8 representa o número de actividades e m=1 o número de projectos.

Crom. = (0.45, 0.44, 0.32, 0.10, 0.08, 0.92, 0.68, 0.70, 0.00, 0.22, 0.26, 0.44, 0.46, 0.89, 0.80, 0.00, 0.00)

Prioridades | Tempos de espera | Datas de libertação

Descodificação das prioridades das actividades

As prioridades das actividades são determinadas a partir dos primeiros n genes de cada cromossoma. O valor da prioridade de uma actividade está compreendido entre 0 e 1. A sua forma de determinação varia de acordo com o algoritmo utilizado.

No algoritmo *GA-SlackMod* a prioridade incorpora o valor da folga da actividade de acordo com a seguinte expressão:

$$Prioridade_j = \frac{Folga_j}{MaiorFolga} \times \left(0.7 + 0.3 \times Gene_j\right)$$

onde

$DE_{i \mid j \in i}$ = 20 unidades de tempo.

$DCML_j$ = Duração do caminho mais longo desde o início da actividade j até ao fim do projecto à qual a actividade j pertence.

$Folga_j$ = $DE_{i \mid j \in i} - DCML_j$

$MaiorFolga$ = 20 unidades de tempo.

A **tab. A.2** apresenta os valores das folgas e das prioridades das actividades, utilizados no exemplo que ilustra o esquema de geração de planos activos parametrizados.

tab.A.2 Valor das prioridades determinadas no exemplo.

$Actividade_j$	$Folga_j$	$Folga_j$ / $MaiorFolga$	$Gene_j$	$PRIORIDADE_j$
0	9	0.45	0.45	0.38
1	10	0.50	0.44	0.37
2	9	0.45	0.32	0.40
3	13	0.65	0.10	0.47
4	14	0.70	0.08	0.51
5	19	0.35	0.68	0.72
6	16	0.80	0.92	0.34
7	20	1.00	0.70	0.91

De seguida a **tab. A.3** apresenta os valores dos tempos de espera de cada iteração, utilizados no exemplo que ilustra o esquema de geração de planos activos parametrizados.

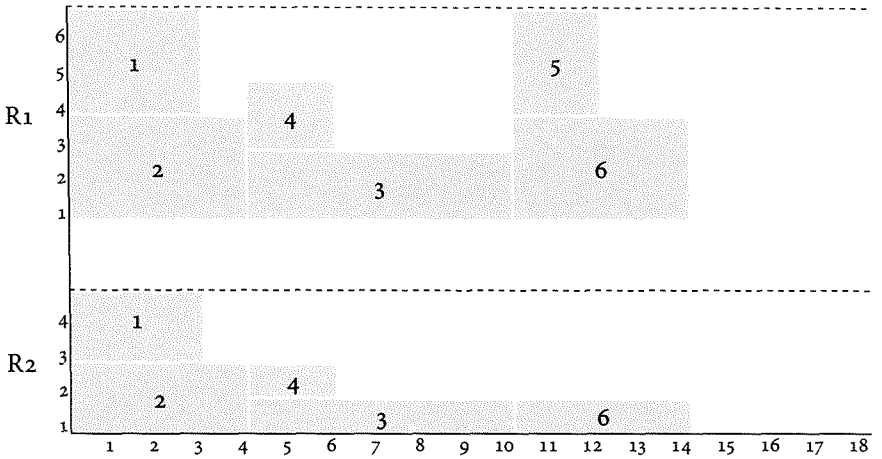
tab.A.3 Valor dos tempos de espera no exemplo.

$Iteração_j$	$Gene_j$	$tEspera_g$
0	0.00	0.00
1	0.22	1.98
2	0.26	2.31
3	0.44	3.96
4	0.46	4.13
5	0.89	8.05
6	0.80	7.25
7	0.00	0.00

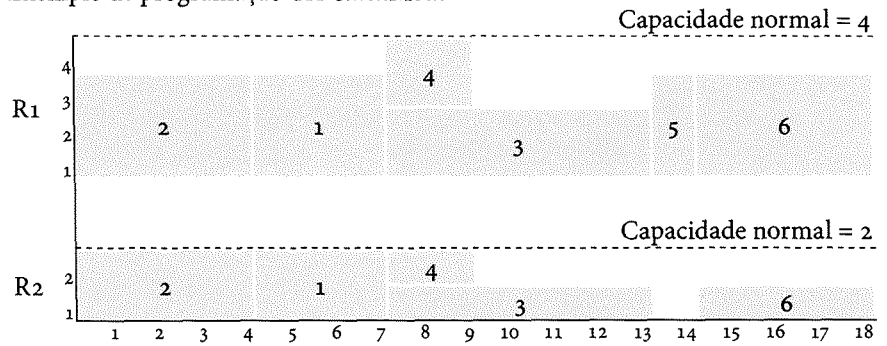
Na **fig. A.8** apresenta-se novamente a solução do problema da **fig. A.7** sob a forma de gráfica de Gantt por aplicação do algoritmo *Ga-SlackMod*.

fig. A.8 Exemplo comparativo de programação de projecto por PERT-CPM e *GA-SlackMod*.

Exemplo de programação PERT-CPM.



Exemplo de programação *GA-SlackMod*.



APÊNDICE B: DESCRIÇÃO DO ALGORITMO GERADOR DE PROBLEMAS PARA O PROBLEMA RCMPSP

Na descrição do algoritmo desenvolvido para a geração de problemas do tipo RCMPSP, utilizar-se-á a seguinte notação:

NP	Número de problemas
NPP	Número de projectos por problema
$NPES$	Número de projectos em execução em simultâneo
DMP	Duração máxima do problema
DIG_i	Data de início gerada do projecto i
NR	Numero de recursos
DE_i	Data de entrega do projecto i
D_i	Duração ideal do projecto i (esta duração corresponde aos melhores valores conhecidos do *makespan*).
$nAct$	Número de actividades.

Na **fig. B.1** apresenta-se o algoritmo gerador de problemas.

fig. B.1 Pseudo-código do algoritmo para gerar problemas.

Inicialização: *NP, NPP, NPS*
$k = 0$
Enquanto $k \leq NP$ **Repetir**
{
  $k = k + 1$
  Para $i = 1$ **até** NPP
  {
    j = Seleccionar aleatoriamente um projecto j do conjunto J120
    $DMP = DMP + D_j$
  }
  $DMP = DMP / NPES$
  Para $i = 1$ **até** NPP
  {
    DIG_i = Inteiro $((DMP - D_i) \times RND)$
    $DE_i = D_i + DIG_i$
    Calcular_Capacidade_Disponível_Recursos() (ver **fig. B.2**)
  }
    Calcular_Capacidade_Mínima_Recursos() (ver **fig. B.3**)
}

Apresenta-se na **fig. B.2** o pseudo-código da função para calcular a capacidade dos recursos e na **fig. B.3** o pseudo-código da função para calcular a capacidade mínima dos recursos.

fig. B.2 Pseudo-código do algoritmo para calcular capacidade dos recursos.

```
Calcular_Capacidade_Disponível_Recursos()

Para k=1 até NR                    (Para todos os recursos)
{
  Para i=1 até NPP                 (Para todos os projectos de cada problema)
  {
    Para t=DIG_i+1 até DE_i        (Desde o início até ao fim do projecto i)
    {
      R_k(t) = R_k(t) + r_j,k | j ∈ i
    }
  }
}
```

fig. B.3 Pseudo-código do algoritmo para calcular capacidade mínima dos recursos.

```
Calcular_Capacidade_Mínima_Recursos()
Calcular mínimo por recurso
Para j=1 até nAct
{
  Para todos os recursos utilizados pela actividade j
  Para k=1 até NR_j
  {
    Determinar mínimo do recurso K
    MR_k=Máximo (r_j,k, MR_k)
  }
}
Atribuir capacidade mínima por recurso ao longo de todo o problema
Para t=1 até DMP
{
  Para k=1 até NR
  {
    Se R_k(t) < MR_k Então R_k(t) = MR_k Fim Se
  }
}
```

Aarts, E.H.L., Van Laarhoven, P.J.M., Lenstra, J.K. e Ulder, N.L.J. (1994). A computational study of local search algorithms for job shop scheduling. *ORSA Journal on Computing*, Vol. 6, pp. 118–125.

Adams, J., Balas, E. e Zawack., D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science*, Vol. 34, pp. 391–401.

Agin, N. (1966). Optimum seeking with branch and bound. *Management Science*, Vol. 13, pp.176–185.

Aiex, R.M., Binato S. e Resende, M.G.C. (2003). Parallel GRASP with Path-Relinking for Job Shop Scheduling. *Parallel Computing*, Vol. 29, pp. 393–430.

Alvarez-Valdés, R. e Tamarit, J.M. (1989). Heuristic Algorithms for Resource – Constrained Project Scheduling: a Review and an Empirical Analisys. In: R. Slowinski e J. Weglarz (ed.). *Advances in Project Scheduling*, Elsevier, Amsterdam, pp. 113–134.

Alvarez-Valdés, R. e Tamarit, J.M. (1993). The Project Scheduling Polyhedron: Dimension, Facets and Lifting Theorems. *European Journal of Operational Research*, Vol. 67, pp. 204–220.

Antill, J.M. e Woodhead, R.W. (1970). *Critical Path Methods in Construction Practice*. Wiley-Interscience, John Wiley & Sons, Inc.. New York.

Applegate, D. e Cook, W. (1991). A computational study of the job-shop scheduling problem. *ORSA Journal on Computing*, Vol. 3, pp. 149–156.

Arora, R.K. e Sachdeva (1989). Distributed Simulation of Resource Constrained Project Scheduling. *Computers & Operations Research*, N.º 16, pp. 295–304.

Ash, R.C. (1999). Activity Scheduling in the Dynamic, Multi-project setting: Choosing Heuristics Through Deterministic Simulation. *Proceedings of the 1999 Winter Simulation Conference*, USA.

Baar, T., Brucker, P. e Knust, S. (1998). Tabu-search Algorithms and lower bounds for the resource-constrained project scheduling problem. In: Voss, S., Martello, S., Osman, I. e Roucairol, C. (eds.). *Meta-heurisitics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer, Boston, pp. 1-8.

Balas, E. (1969). Machine Scheduling via Disjunctive Graphs: An Implicit Enumeration Algorithm. *Operations Research*, Vol. 17, pp. 941–957.

Bandelloni, M., Tucci, M. e Rinaldi, R. (1994). Optimal Resource Leveling Using Non-Serial Dynamic Programming. *European Journal of Operational Research*, Vol. 78, pp. 162–177.

Baker, K.R. (1974). *Introduction to Sequencing and Scheduling*. John Wiley, New York.

Baker, K.R. e Bertrand, J.W.M. (1981). An Investigation of Due Date Assignment with Constrained Tightness. *Journal of Operations Management*, Vol. 1, N.º 3, pp. 109–120.

Baker, K.R. e Schrage, L.E. (1978). Finding an optimal sequence by dynamic programming: an extension to precedence related tasks. *Operations Research*, Vol. 26, pp. 111–120.

Barker, J.R. e McMahon, G.B. (1985). Scheduling the general job shop. *Management Science*, Vol. 31, pp. 594–598.

Barnes, J.W. e Chambers, J.B. (1995). Solving the job shop scheduling problem using tabu search. *IIE Transactions*, Vol. 27, pp. 257–263.

Bean, J.C. (1994). Genetics and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing*, Vol. 6, pp. 154–160.

Beasley, D., Bull, D.R. and Martin, R.R. (1993). An Overview of Genetic Algorithms: Part 1, Fundamentals, University Computing, Vol. 15, N.º 2, pp. 58-69, Department of Computing Mathematics, University of Cardiff, UK.

Beasley, J.E. e Chu, P.C. (1996). A Genetic Algorithm for the Set Covering Problem. *European Journal of Operational Research*, Vol. 94, pp. 392–404.

Belchior, P.G.O.B. (1970). PERT/CPM – *Técnica de Avaliação, Revisão e Controle de Projetos*. Edições de Ouro, Brasil.

Bell, C.E. e Park, K. (1990). Solving Resource Constrained Project Scheduling Problems by A* Search. *Naval Res. Logist*, N.º 37, pp. 61–84.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

Biegal, J.E. e Davern, J.J. (1990). Genetic Algorithm and Job Shop Scheduling. *Computers and Industrial Engineering*, Vol. 19.

Binato, S., Hery, W.J., Loewenstern, D.M. e Resende, M.G.C. (2002). A GRASP for Job Shop Scheduling. In: Ribeiro, C.C., Hansen, P. (eds.). *Essays and Surveys in Metaheuristics*, Kluwer Academic Publishers.

Blackstone Jnr, J.H., Phillips, D.T. e Hogg, G.L. (1982). A State-of-art Survey of Dispatching Rules for Manufacturing Shop Operations. *International Journal of Production Research*, Vol.20, N° 1, pp. 27–45.

Blazewicz, J., Lenstra, J.K., e Rinnooy Kan, A.H.G. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics*, N.º 5, pp. 11–24.

Boctor, F.F. (1990). Some Efficient Multi-Heuristics Procedures for Resource-Constrained Project Scheduling. *European Journal of Operational Research*, Vol. 49. pp. 3–13.

Boctor, F.F. (1996a). A New and Efficient Heuristic for Scheduling Projects with Resource Restrictions and Multiple Execution Modes. *European Journal of Operational Research*, Vol. 90. pp. 349–361.

Boctor, F.F. (1996b). Resource-constrained project scheduling simulated annealing. *International Journal of Production Research*, Vol. 34(8), pp. 2335–2354.

Bouleimen, K. e Lecocq, H. (2003). A new efficient simulated annealing Algorithms for the resource-constrained project scheduling problem. *European Journal of Operational Research*, Vol. 149, pp. 268–281.

Bouleimen, K. e Lecocq, H. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, Vol. 149(2), pp. 268–281.

Bowman, E.H. (1959). The schedule - sequencing problem. *Operations Research*, Vol. 7, pp. 621–624.

Boyd, E.A. e Burlingame, R. (1996). *A Parallel Algorithm for Solving Difficult Job-Shop Scheduling Problems.* Operations Research Working Paper, Department of Industrial Engineering, Texas A&M University, College Station, Texas, USA pp. 77843–3131.

Brandt, J.D., Meyer, W.L. e Shaffer, L.R. (1964). The Resouce Scheduling Problem in Construction. *Civil Engineering Studies Report,* N.° 5, Department of Civil Engineering, University of Illinois.

Brindle, A. (1981). *Genetics algorithms for function optimization.* Unpublished doctoral dissertation, University of Alberta, Edmonton.

Brooks, G.H. e White, C.R. (1965). An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem. *Journal of Industrial Engineering*, January-February Issue, pp. 34–40.

Brucker, P., Jurisch, B. e Sievers, B. (1994). A Branch and Bound Algorithm for Job-Shop Scheduling Problem, *Discrete Applied Mathematics*, Vol. 49, pp. 105–127.

Brucker, P., Jurisch, B. e Kramer, A. (1992). *The job shop problem and immediate selection*, Osnabrucker Schriften zur Mathematik, Fachbereich Mathematik, Universitat Osnabruck.

Brucker, P. e Knust, S. (2000). A linear programming and constraint propagation-based lower bound for the RCPSP. *European Journal of Operational Research*, Vol. 127, pp. 355–362.

Brucker, P., Knust, S. Schoo, A. e Thiele, O. (1998). A branch and bound Algorithms for the resource-constrained project scheduling problem. *European Journal of Operational Research*, Vol. 107, pp.272–288.

Bruns, R. (1993). Direct chromosome representation and advanced genetic operators for production scheduling. *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, California, pp. 352–359.

Buffa, E.S. e Sarin, R.K. (1987). *Modern Production/ Operations Management.* John Wiley & Sons.

Callahan, M.T. Quackenburh, D.G. e Rowings, J.E. (1992). *Construction Project Scheduling.* MacGraw-Hill.

Carlier, J. (1982). The one-machine sequencing problem. *European Journal of Operational Research*, Vol. 11, pp. 42–47.

Carlier, J. (1987). Scheduling jobs with release dates and tails on identical machines to minimize makespan. *European Journal of Operational Research*, pp. 298–306.

Carlier, J. e Latapie, B. (1991). Une methode arborescente pour resoudre les problemes cumulatifs, *Recherche operationalle*, Vol. 25, pp. 311–340.

Carlier, J. e Pinson, E. (1989). An Algorithm for Solving the Job Shop Problem. *Management Science*, Feb., Vol. 35, pp.164–176.

Carlier, J. e Pinson, E. (1990). A practical use of Jackson's preemptive schedule for solving the job-shop problem. *Annals of Operations Research*, Vol. 26, pp. 269–287.

Chambers, L. (1995). *Practical handbook of genetic algorithms: Applications.* CRC Press, Boca Raton.

Cheng, R., Gen, M. e Tsujimura, Y. (1999). A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies, *Computers & Industrial Engineering*, Vol. 36, pp. 343–364.

Christofides, N. Alvarez-Valdés, R. Tamarit, J.M. (1987). Problem scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, Vol. 29, pp. 262–273.

Cleveland, G.A. e Smith, S.F. (1989). Using Genetic Algorithms to Schedule Flow Shop Release In: *Proceedings of the 3.rd International Conference on Genetic Algorithms Applications*, pp. 160–169.

Cooper, D.F. (1976). Heuristics for Scheduling Resource – Constrained Projects: An Experimental Investigation. *Management Science*, Vol. 22, pp. 1186–1194.

Conway, R.W. (1984). *A User's Guide to the Factory Modeling System.* TR 84–596, Department of Computer Science, Cornell University, Ithaca, New York.

Conway, R.W., Maxwell, W.L. e Miller, L.W. (1967). *Teory of Scheduling.* Addison-Wesley, Reading, Mass.

Corwin, B.D. e Esogbue, A.O. (1974). Two machine flow shop scheduling problems with sequence dependent set up times: a dynamic programming approach. *Naval Res. Logist. Quar*, Vol. 21, pp. 515–524.

Croce, F., Menga, G., Tadei, R., Cavalotto, M. e Petri, L. (1993). Cellular Control of Manufacturing Systems, *European Journal of Operations Research*, Vol. 69, pp. 498–509.

Croce, F., Tadei, R. e Volta, G. (1995). A Genetic Algorithm for the Job Shop Problem, *Computers and Operations Research*, Vol. 22(1), pp. 15–24.

Dannenbring, D.G. (1977). An evaluation of flow shop sequencing heuristics. *Management Science*, Vol. 23, pp. 1174–1182.

Dantzig, G.B. (1960). A machine shop scheduling model. *Management Science*, Vol. 6, pp.191–196.

Davis, L. (1985). Job shop scheduling with genetic algorithms. In: *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, Morgan Kaufmann, pp. 136–140.

Davis, L. (1989). Adapting Operator Probabilities in Genetic Algorithms. *Proceeding of The Third International Conference on Genetic Algorithms*. San Mateo, CA, pp. 61–69.

Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, Princeton, N.J.

Davis, W.E. (1973). Project Scheduling Under Resource Constraints – Historical Review and Categorization of Procedures. *AIIE Transactions*, Vol. 5, N.º 4, pp. 147–163.

Davis, W.E. e Patterson, J.H. (1975). A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling. *Management Science*, Vol. 21, N.º 8, pp. 944–955.

De Jong, K.A. (1985). *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.

De Jong, K.A. e Spears, W. (1990). An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms. *Proceedings of the 1st International Conference on Parallel Problem Solving from Nature*, Dortmund, Germany.

Deckro, R.F., Winkofsky, E.P., Hebert, J.E., e Gagnon R. (1991). A Decomposition Approach to multi-project scheduling. *European Journal of Operational Research*, Vol. 51, pp. 110–118.

Dell'Amico, M. e Trubian, M. (1993). Applying tabu search to the job shop scheduling problem. *Annals of Operations Research*, Vol. 41, pp. 231–252.

Demeulemeester, E. (1992). *Optimal Algorithms for Various Classes of Multiple Resources-constrained Project Scheduling Problems*, Ph.D. Tesis, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.

Demeulemeester, E. e Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem, *Management Science*, Vol. 38, pp. 1803–1818.

Demeulemeester, E. e Herroelen, W. (1997). New benchmark results for the resource-constrained project scheduling problem. *Management Science*, Vol. 43, pp. 1485–1492.

Doersch, R.H. e Patterson, J.H. (1977). Scheduling a project to maximize its present value: a zero-one programming approach. *Management Sciences*, Vol. 23(8), pp. 882–889.

Dorndorf, U. e Pesch, E. (1995). Evolution Based Learning in a Job Shop Environment. *Computers and Operations Research*, Vol. 22, pp. 25–40.

Dorndorf, U., Pesch, E. e Phan Huy, T. (2000a). A branch-and-bound algorithm for the resource-constrained project scheduling problem. *Mathematical Methods of Operations Research*, N.º52, pp. 413–439.

Dorndorf, U., Pesch, E. e Phan Huy, T. (2000b). A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalized precedence constraints. *Management Science*, N.º46, pp. 1365–1384.

Drexl, A. (1991). Scheduling of Project Networks by Job Assignment. *Management Science*, N.º 37, pp. 1590–1602.

Dumond, J. e Mabert, Vol.A. (1988). Evaluating Project Scheduling And Due Date Assignment Procedures: An Experimental Analysis. *Management Science*, Vol. 34, n. 1, pp. 101–118.

Easa, S.M. (1989). Resource Leveling in Construction by Optimization. *Journal of Construction Engineering and Management*, N.º 115(2), pp. 302–316.

Elmaghraby, S.E.(1977). *Activity Networks: Project Planning and Control by Network Models*, Wiley, New York.

Elsayed, E.A. (1982). Algorithms for Project Scheduling with Resource Constraints. *International Journal of Production Research*, N.º 20, pp. 95–103.

Eshelman, L.J. e Schaffer, J.D. (1991). Preventing Premature Convergence in genetic Algorithms by Preventing Incest. In Belew, R.K. e Booker, L.B. (eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann.

Falkenauer, E. e Bouffouix, S. (1991). A Genetic Algorithm for Job Shop. *Proceedings IEEE International Conference on Robotics and Automation*. Los Alamitos, California, USA.

Fang, H.L., Ross,P. e Crone, D. (1993). A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 375–382

Fendley, L.G. (1968). Towards the Development of a Complete Multiproject Scheduling System. *Journal of Industrial Engineering*, pp. 505–515.

Feng, C.W, Liu, L., e Burns, S.A. (1997). Using Genetic Algorithms to Solve Construction Time-Cost Trade-Off Problems. *Journal of Computing Engineering*, Vol. 11, N.º 3, pp. 184–190.

Ficici, S.G., Melnik, O. e Pollack, J.B. (2000). A Game-Theoretic Investigation of Selection Methods Used in Evolutionary Algorithms. In: Zalgala, A *et al.* (eds.). *Proceedings of the Congress on Evolutionary Computation*, IEEE Press 2000.

Fisher, H. e Thompson, G.L. (1963). Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules. In: Muth, J.F. e Thompson, G.L. (eds.). *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, NJ, pp. 225–251.

Florian, M., Trépant, P. e McMahon, G. (1971) An Implicit Enumeration Algorithm for the Machine Sequencing Problem. *Management Science Application Series*, August, N.º 17(12), pp. 782–792.

Fourman, M.P. (1985). Compaction of Symbolic Layout Using Genetic Algorithms. *Proc. Int. Conf. Genetic Algorithms Appl.*, pp. 141–153.

Fox, B.R. e McMahon, M.B. (1991). Genetic Operators for Sequencing Problems. *Foundations of Genetic Algorithms*, Morgan Kaufmann, pp.284–300

French, S. (1982). *Sequencing and Scheduling – An Introduction to the Mathematics of the Job-Shop*, Ellis Horwood, John – Wiley & Sons, New York.

Gauthier, F.A.O. (1983). *Programação da Produção: uma Abordagem Utilizando Algoritmos Genéticos*. Tese de Doutorado, UFSC, Florianópolis.

Gere, W.S. Jr. (1966). Heuristics in job-shop scheduling. *Management Science* N.º 13, pp.167–190.

Giffler, B. e Thompson, G.L. (1960). Algorithms for Solving Production Scheduling Problems. *Operations Research*, Vol. 8(4), pp. 487–503.

Glover, F. (1989). Tabu Search: Part I. *ORSA Journal on Computing*, Vol. 1, pp. 190–206.

Glover, F. (1990). Tabu Search: Part II. *ORSA Journal on Computing*, Vol. 2, pp. 4–12.

Goldberg, D.E. (1984). Computed-Aided Pipeline Operation Using Genetic Algorithms and Rule Learning. *API Pipeline Cybernetics Symp.*, Houston, Texas, TX.

Goldberg, D.E. (1987). *A Note on The Disruption Due to Crossover in a Binary-coded Genetic Algorithm (Rcga Report N.º87001)*, Tuscaloosa: University of Alabama, The Clearinghouse for Genetic Algorithms.

Goldberg, D.E. (1989). *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley.

Goldberg, D.E. (1990). A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-oriented Simulated Annealing. *Complex Systems*, N.º 4, pp. 445–460.

Goldberg, D.E. e Lingle, Jr. R. (1985). Alleles, Loci and the Travelling Salesman Problem. *Proceedings of an Carnegie-Melon University*, Pittsburgh, Pennsylvania, pp. 154–159.

Gonçalves, J.F. e Beirão, N.C. (1999). Um Algoritmo Genético Baseado em Chaves Aleatórias para Sequenciamento de Operações. *Revista Associação Portuguesa de Desenvolvimento e Investigação Operacional*, Vol. 19, pp. 123–137.

Gonçalves, J.F. e Mendes, J.M. (1994). A Look-Ahead Dispatching Rule for Scheduling Operations. *VII Latin-Ibero-American Conference on Operations Research and Systems Engineering*, University of Chile, Santiago, Chile.

Gray, C. e Hoesada, M. (1991). Matching Heuristic Scheduling Rules for Job Shops to the Business Sales Level. *Production and Inventory Management Journal*, Vol. 4, pp. 12–17.

Grefenstette, J.J. (1986). Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 16, N.º 1, pp. 122–128.

Grefenstette, J.J., Gopal, R., Rosmaita, B., Van Gucht, D. (1985). Genetic Algorithm for the Traveling Salesman Problem. *Proceedings of an Carnegie-Melon University*, Pittsburgh, Pennsylvania, pp. 160–169.

Harris, F., Mccaffer, R. (1977). *Modern Construction Management*, Crosby Lockwood Staples, London, UK.

Harding, H.A. (1981). *Administração da Produção*. Editora Atlas, São Paulo, Brasil.

Hartmann, S. (1998). A competitive genetic algorithms for resource-constrained project scheduling. *Naval Research Logistics*, Vol. 45, pp. 733–750.

Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics*, Vol. 49, pp. 433–448.

Hartmann, S. e Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, Vol. 127, pp. 394–407.

Hartmann, S. e Sprecher, A. (1996) Hierarchical models for multi-project planning and scheduling. *European Journal of Operational Research*, Vol. 94, pp. 377–383.

Hastings, N.A.J. (1972). On Resource Allocation in Project Networks. *Journal of the Operational Research Society*, Vol. 23, pp. 217–221.

Held, M. e Karp, R.M. (1962). A dynamic programming approach to sequencing problems. *Proceedings of the 16th ACM national meeting*, pp. 71.201–71.204.

Heller, J. e Logemann, G. (1962). An Algorithm for the Construction and Evaluation of Feasible Schedules. *Management Science*, Vol. 8, pp. 168–183.

Herroelen, W., DE Reyck, B. e Demeulemeester, E. (1998). Resource-constrained project scheduling: a survey of recent developments, *Computers and Operations Research*, Vol. 25(4), 279–302.

Hoitomt, D.J., Luh, P.B. e Pattipati, K.R. (1993). Practical approach to job-shop scheduling problems. *IEEE Trans. Rob. Autom*, 9/1, Feb., pp. 1–13.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Holloway, C.A. e Nelson, R.T. (1974). Job Shop Scheduling With Due Dates and Variable Processing Times. *Management Science*, Vol. 20, N. 9, pp.1264–1275.

Hollstien, R.B. (1971). *Artificial genetic adaptation in computer control systems*. Tese de Doutoramento, University of Michigan. Dissertation Abstracts International, 32 (3), 1510B. University Microfilms N.º 71–23, 773).

Ichihara, J. (1997). *Um Método de Solução Heurístico para a Programação de Edifícios Dotados de Múltiplos Pavimentos-Tipo*. Tese de Doutorado. UFSC, Florianópolis.

Icmeli, O., Erenguc, S.S., Zappe, C.J. (1993). *Project scheduling problems: A survey, International Journal of Operations & Production Management* Vol. 13 (11), pp. 80–91.

Icmeli, O, e Rom, W.O. (1997). Ensuring quality in resource constrained project scheduling. *European Journal of Operational Research*, Vol. 107, pp. 431–450.

Jain, A.S. e Meeran, S. (1999). A State-of-the-Art Review of Job-Shop Scheduling Techniques. *European Journal of Operations Research*, Vol. 113, pp. 390–434.

Jain, A.S., Rangaswamy, B. e Meeran, S. (1998). *New and Stronger Job-Shop Neighborhoods: A Focus on the Method of Nowicki and Smutnicki (1996)*. Department of Applied Physics, Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland.

Janikow, C.Z. e Michalewicz, Z. (1991). An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms. In: Belew R.K. e Booker, L.B. (eds.). *Proceedings of the 4th International Conference on Genetic Algorithms*, Morgan Kaufmann.

Johnson, S.M., (1954). Optimal Two and Three-Stage Production Schedules with Set-Up Times Included. *Naval Research Logistics Quarterly*, Vol. 1, pp. 61–68.

Johnson, T.J.R. (1967). *An algorithm for the resource-constrained project scheduling problem*, Dissertation, Massachusetts Institute of Technology, USA.

Kaplan, L.A. (1988). *Resource Constrained Project Scheduling with Preemption of Jobs*, Unpublished Ph.D. Tesis, University of Michigan.

Kapsalis, A., Rayward-Smith V.J. e Smith G. D. (1993). Solving the Graphical Steiner Tree Problem Using Genetic Algorithms. *Journal of the Operational Research Society*, Vol. 44(4), pp. 397–406.

Kelley, J.E. Jr., (1963). The Critical-Path Method: Resources Planning and Scheduling. In: Muth, J.F. e G.L. Thompson (eds.). *Industrial Scheduling, Prentice Hall, Englewood Cliffs*, pp. 347–365.

Kise, H., Ibraki, T. e Mine, H. (1978). A solvable case of the one machine sche-

duling problem with ready times and due dates. *Operations Research*, Vol. 26, pp.121–126.

Klein, R. (1998). Bidirectional planning – improving priority rule based heuristics for scheduling resource-constrained projects. *Technical Report 10/98, Schriften zur Quamtitativen Betriebswirtschaftslehre*, Technische Universitat Darmstadt.

Klein, R. (2000). *Scheduling of Resource-Constrained Projects*, Kluwer Academic Publishers, Boston.

Klein, R. e Scholl, A. (1998a). *Progress: Optimally solving the generalized resource-constrained project scheduling problem.* Working paper, University of Technology, Darmstadt.

Klein, R. e Scholl, A. (1998b). *Scattered branch and bound: An adaptative search strategy applied to resource-constrained project scheduling problem.* Working paper, University of Technology, Darmstadt.

Kobayashi, S., Ono, I., Yamamura, M. e Syswerda, G. (1995). An Efficient Genetic Algorithm for Job-Shop Scheduling Problems. *Proceedings of the 6th International Conference on Genetic Algorithms*, pp.506–511

Kohlmorgen, U., Schmeck, H. e Haase, K. (1999). Experiences with fine-grained parallel genetic algorithms. *Annals of Operations Research*, Vol. 90, pp.203–219.

Kolisch, R. (1995). *Project Scheduling under Resource Constraints.* Physica-Verlag Heidelberg, Germany.

Kolisch, R. (1996a). Efficient Priority Rules for the Resource-Constrained Project Scheduling Problem. *Journal of Operations Management*, Vol. 14, pp. 179–192.

Kolisch, R. (1996b). Serial and Parallel Resource-Constrained Project Scheduling Methods Revisited: Theory and Computation. *European Journal of Operational Research*, Vol. 90, pp. 320–333.

Kolisch, R. e Drexl, A (1996). Adaptative search for solving hard project scheduling problems. *Naval Research Logistics*, Vol. 43, pp. 23–40.

Kolisch, R. e Hartmann, S. (1999). Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. In: Weglarz, J. (ed.), *Project Scheduling: Recent Models, Algorithms and Applications*, Cap. 7, Dordrecht, Kluwer, pp. 147–178.

Kolisch, R., Schwindt, C. e Sprecher, A. (1998). Benchmark instances for project scheduling problems. In: Weglarz, J. (ed.). *Handbook on recent advances in project scheduling*, Cap. 9, Dordrecht, Kluwer.

Kolisch, R. e Padman, R. (2001). An integrated survey of deterministic project scheduling. *The International Journal of Management Science*, Vol. 29, pp. 249–272.

Kolonko, M. (1999). Some New Results on Simulated Annealing Applied to the Job Shop Scheduling Problem. *European Journal of Operational Research*, Vol. 113, pp. 123–136.

Krüger, K., Shakhlevich, N.V., Sotskov, Y.N. e Werner, F. (1995). A Heuristic Decomposition Algorithm for Scheduling Problems on Mixed Graphs. *Journal of the Operational Research Society*, Vol. 46, pp. 1481–1497.

Kurtulus, I.S., Davis, E.W. (1982). Multi-project scheduling: Categorization of heuristic rules performance. *Management Science*, Vol. 28 (2), pp. 161–172.

Kurtulus, I.S., Narula, S.C., (1985). Multi-Project Scheduling: Analysis Of Project Performance. *IIE Transactions*, Vol. 17 (1), pp. 58–66.

Laarhoven, P.J.M.V., Aarts, E.H.L. e Lenstra, J.K. (1992). Job Shop Scheduling By Simulated Annealing. *Operations Research*, Vol. 40, pp. 113–125.

Lawer, E.L. (1973). Optimal Sequencing of a Single Machine Subject to Precedence Constaints. *Management Science*, Vol. 19, pp. 544–546.

Lawer, E.L. e Moore, J.M. (1969). A functional equation and its application to resource allocation and sequencing problems. *Management Science*, Vol. 16, pp. 77–84.

Lawer, E.L. e Wood, D.E. (1966). Branch and bound methods: a survey. *Operations Research*, Vol. 14, pp.1098–1112.

Lawrence, S. (1984). *Resource-Constrained Project Scheduling: A Computational Comparison of Heuristic Scheduling Techniques.* Working Paper, GSIA, Carnegie Mellon University, Pittsburgh, PA.

Lawrence, S.R. e Morton, T.E. (1993). Resource-Constrained Multi-Project Scheduling with Tardy Costs: Comparing Myopic Bottleneck and Resource Pricing Heuristics. *European Journal of Operational Research*, Vol. 64, pp. 168–187.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. e Shmoys, D.B. (1993). Sequencing and Scheduling: Algorithms and Complexity. In: Graves, S.C., Rinnooy Kari, A.H.G. e Zipkin, P.H. (eds.). *Handbooks in Operations Research and Management Science*, Vol. 4, Logistics of Production and Inventory, Elsevier, Amsterdam.

Leachman, R.C. (1993). Multi-Project Scheduling With Explicit Lateness Costs. *IIE Transactions*. Vol. 25, N.º 2, pp. 34–44.

Lee, J.K. e Kim, Y.D. (1996). Search heuristics for resource constrained project scheduling. *Journal of the Operational Research Society*, Vol. 47, pp. 678–689.

Lenstra, J.K. (1977). *Sequencing by Enumerative Methods.* Mathematisch Centrum, Amsterdam.

Lenstra, J.K., Rinnoy Kan, A.H.G. (1979). Computational complexity of discrete optimisation problems. *Annals of Discrete Mathematics*, Vol. 4, pp. 121–140.

Leon, V.J. e Balakrishnan, R. (1995). Strength and Adaptability of Problem-Space Based Neighborhoods for Resource Constrained Scheduling, *Operations Research Spektrum*, Vol. 17, pp. 173–182.

Levitin, G. e Rubinovitz, J. (1993). Genetic Algorithm for Linear and Cyclic Assignment problem. *Computers and Operations Research*, Vol. 20, N.º 6, pp. 597–586.

Li, K.Y. e Wills, R.J. (1992). An Iterative Scheduling Technique for Resource-Constrained Project Scheduling. *European Journal of Operational Research*, Vol. 56, pp. 370–379.

Liepins, G.E. e Hilliard, M.R. (1989). Genetic Algorithms: Foundations and Applications. *Annals of Operations Research*, Vol. 21, pp. 31–58.

Lourenço, H.R. (1995). Local optimization and the job-shop scheduling problem. *European Journal of Operational Research*, Vol. 83, pp. 347–364.

Lourenço, H.R. e Zwijnenburg, M. (1996). Combining the large-step optimization with tabu-search: Application to the job-shop scheduling problem. In: Osman I.H. e Kelly J.P. (eds.). *Metaheuristics: Theory and Apllications*. Kluwer Academic Publishers, pp. 219–236.

Lova, A. (1997). *Programacion de multiproyetos com recursos limitados: un enfoque multicritério*. Tese de Doutoramento. Universidade Politécnica de Valência.

Lova, A., Maroto, C. e Tormos, P. (2000). A Multicriteria Heuristic Method to Improve Resource Allocation in Multiproject Scheduling. *European Journal of Operational Research*, Vol. 127, pp. 408–424.

Maccarthy, B. L. e Liu, J. (1993). Addressing the Gap Scheduling Research: a Review of Optimization and Heuristic Method in Production Scheduling. *International Journal of Production Research*. Vol. 31, N.º 1, pp. 59–79.

Martin, P.D. (1996). *A Time-Oriented Approach to Computing Optimal Schedules for the Job-Shop Scheduling Problem*. Ph.D. Thesis, School of Operations Research & Industrial Engineering, Cornell University, Ithaca, New York 14853–3801, August.

McMahon, G.B. e Florian, M. (1975). On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness. *Operations Research*, May-June, Vol. 23(3), pp.475-482.

Mayne, A.S. (1960). On the job shop scheduling problem. *Operations Research*, Vol. 8, pp.219-223.

Mattila, K.G. e Abraham, D.M. (1998). *Resource Leveling of Linear Schedules Using Linear Programming*. Vol. 124, N.º 3, pp. 232–244.

Mcmahon, G. e Florian, M. (1975). On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness. *Operations Research*, Vol. 23, N.º 3, pp. 475–482.

Mellor, P. (1966). A Review of Job Shop Scheduling. *Operational Research Quarterly*, Vol. 17, N.º 2, pp. 167.

Mendes, J.J.M. (2003). *Sistema de Apoio à Decisão para Planeamento de Sistemas de Produção do Tipo Projecto*. Tese de Doutoramento. Departamento de Enge-

nharia Mecânica e Gestão Industrial, Faculdade de Engenharia da Universidade do Porto, Portugal.

Mendes, J.M. e Gonçalves, J.F. (2003). Um Algoritmo Genético para o Problema do Sequenciamento de Projectos com Recursos Limitados. *Revista da Associação Portuguesa de Desenvolvimento e Investigação Operacional*, Vol. 23, N.º 2.

Mendes, J.J.M., Gonçalves, J.F. e Brito, A (2003). Um Algoritmo Genético para o Problema do Sequenciamento de Múltiplos Projectos com Recursos Limitados. *Proceedings of VII Congresso de Mecânica Aplicada e Computacional*, Vol. II, Universidade de Évora, Abril 2003, pp. 989–1002.

Meyer, W.L. e Shaffer, L.R. Extensions of the critical path method through the application of integer programming. *Civil Engineering Construction Research Series*, N.º 2, University of Illinois, Urbana, III.

Metropolis, N., Rosenbluth, A., Teller, A, e Teller, E. (1953). Equation of State Calculations by Fast Computing Machines, *Journal of Chemical Physics*, Vol. 21, pp.1087-1092.

Minciardi, R., Paolucci, M. e Puliafito, P.P. (1994). Development of a heuristic project scheduler resource constraints. *European Journal of Operational Research*, Vol. 79, pp. 176–182.

Mingozzi, A., Maniezzo, V., Ricciardelli, S. e Bianco, L. (1998). An Exact Algorithm for Project Scheduling with Resource Constraints Based on a New Mathematical Formulation. *Management Science*, Vol. 44, pp. 714–729.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT, Cambridge.

Moder, J.J., Phillips, C.R. e Davis, E.W. (1983). *Project Management With CPM, PERT and Precedence Diagramming*. 3.a ed., Reinhold, New York.

Mohanthy, R.P. e Siddiq, M.K. (1989). Multiple Projects Multiple Resources-Constrained Scheduling: Some Studies. *International Journal of Production Research*. Vol. 27, N.º 2, pp. 261–280.

Mohring, R.H., Schulz, A.S., Stork, F. e Uetz, M. (2002). *Solving Project Scheduling Problems by Minimum Cut Computations. MIT Sloan School of Management.*Working Paper 4231–02, April 2002, Massachusetts, USA.

Nakano, R. (1991). Conventional Genetic Algorithm for Job Shop Problems. *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 474–479.

Neppalli V.R., Chen C.L. e Gupta J.N.D. (1996). Genetic Algorithms for the Two-Stage Bicriteria Flow Shop Problem. *European Journal of Operational Research*, Vol. 95, pp. 356–373.

Nonobe, K. e Ibaraki, T. (2001). Formulation and Tabu Search Algorithm for the Resource Constrained Project Scheduling Problem. In: Ribeiro, C.C. e Hansen P. (eds.). *Essays and Surveys in Meta-heuristics*, Cap. 25, Dordrecht, Kluwer, pp. 557–588.

Norman, B. e Bean, J. (1997). Random Keys Genetic Algorithm for Job Shop Scheduling. *Engineering Design and Automation*, Vol. 3, pp. 145–156.

Nowicki, E. e Smutnicki, C. (1996). A Fast Taboo Search Algorithm for the Job-Shop Problem, *Management Science*, Vol. 42, N.º 6, pp. 797–813.

Oguz, O. e Bala, H. (1994). A comparative study of computational procedures for the resource constrained project scheduling problem. *European Journal of Operational Research*, Vol. 72, pp. 406–416.

Oliveira, J.A.Vol. (2000). *Aplicação de Modelos e Algoritmos de Investigação Operacional ao Planeamento de Operações em Armazéns*. Tese de Doutoramento, Universidade do Minho.

Oliver, I.M., Smith, D.J. e Holland, R. C. (1987). A Study of Permutation Crossover Operations on the Travelling Salesman Problem. *Proceedings of the 2nd International Conference on Genetic Algorithms*, pp. 224–230.

O'Reilly, U.M. e Oppacher, F. (1995). The Troubling Aspects of a Building Block Hypothesis for Genetic Programming. In: Whitley L.D. e Vose M.D. (eds.). *Foundations of Genetic Algorithms*, Vol. 3, Morgan Kaufmann.

Özdamar, L. e Ulusoy, G. (1994). A Local Constraint Based Analysis Approach to Project Scheduling Under General Resource Constraints. *European Journal of Operational Research*, Vol. 79, pp. 287–298.

Özdamar, L. e Ulusoy, G. (1996a). A Note on an Iterative Forward/Backward Scheduling Technique with Reference to a Procedure by Li and Willis. *European Journal of Operational Research*, Vol. 89, pp. 400–407.

Özdamar, L. e Ulusoy, G. (1996b). An Iterative Local Constraint Based Analysis for Solving the Resource Constrained Project Scheduling Problem. *Journal of Operations Management*, Vol. 14(3), pp. 193–208.

Palmer, D.S. (1965). Sequencing jobs through a multi-stage process in the minimum total time – a quick method of obtaining a near optimum. *Operational Research Quarterly*, Vol. 16, pp. 101–107.

Pascoe, T.L. (1966). Allocation of Resources CPM. *Revue Francaise Recherche Operationalle*, Vol. 38, pp. 31–38.

Panwalkar, S.S e Iskander, W. (1977). A Survey of Scheduling Rules. *Operations Research*, Vol. 25, N.º 1, pp. 45–61.

Patterson, J.H. (1973). Alternate Methods of Project Scheduling With Limited Resources. *Naval Research Logistics Quarterly*, Vol. 20, pp. 767–784.

Patterson, J.H. (1976). Project Scheduling: the Effects of Problem Structure on Heuristic Performance. *Naval Research Logistics Quarterly*, Vol. 23, pp. 95–123.

Patterson, J.H. (1984). A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem. *Management Science*, Vol. 30, pp. 854–867.

Patterson, J.H. e Huber, W.D. (1974). A Horizon-Varying, Zero-One Approach to Project Scheduling. *Management Science*, Vol. 20, N.º 6, pp. 990–998.

Patterson, J.H., Slowinski, R., Talbot, F.B. e Weglarz, J. (1989). An Algorithm for a General Class of Precedence and Resource Constrained Scheduling Problems. In: Slowinski, R. e Weglarz, J. (eds.). *Advances in Project Scheduling*, Elsevier, Amsterdam, pp. 3–28.

Payne, J.H. (1995). Management of multiple simultaneous projects: A state-of-the-art review. *International Journal of Project Management*. Vol. 13, pp. 163–168.

Perregaard, M. e Clausen, J. (1995). *Parallel Branch-and-Bound Methods for the Job-shop Scheduling Problem*. Working Paper, University of Copenhagen, Copenhagen, Denmark.

Pinson, E. (1995). The job shop scheduling problem: A concise survey and some recent developments. In: Chrétienne, P., Coffman Jr., E.G., Lenstra, J.K. e Liu, Z. (eds.). *Scheduling theory and its application*, John Wiley & Sons, Inc., pp. 277–293.

Pinson, E., Prins, C. e Rullier, F. (1994). Using tabu search for solving the resource-constrained project scheduling problem. *Proceedings of the 4th International Workshop on Project Management and Scheduling*, Leuven, pp. 102–106.

Pritsker, A., Allan, B., Watters, L.J. e Wolfe, P.M. (1969). Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, Vol. 16(1), pp. 93–108.

Reeves, C. R. (1991). An Introduction to Genetic Algorithms. In: Munford, A.G. e Bailey, T.C. (eds.). *Operational Research Society*, Operations Research Tutorial Papers, pp. 69–83.

Reeves, C. R. (1993). *Modern Techniques for Combinatorial Problems*. John Wiley & Sons, Inc. New York.

Reeves, C. R. (1995). A Genetic Algorithm for Flowshop Sequencing. *Computers Operations Research*, Vol. 22, N.º 1, pp. 5–13.

Reiss, G. (1996). Multi-Project Scheduling and Management. *Project – the magazine of the UK Association for Project Management*. ISSN: 0957–7033. Vol. 9. Dec 96 & Jan 97

Resende, M.G.C. (1997). A GRASP for Job Shop Scheduling. *INFORMS Spring Meeting*, San Diego, California, USA.

Reyck, B.D. e Herroelen, W. (1998). A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, Vol. 111, pp. 152–174.

Rinnooy Kan, A.H.G. (1976). *Machine Scheduling Problems: Classification, Complexity and Computations*. Martinus Nijhoff, The Hague, Holland.

Roy, B. e Sussmann, B. (1964). *Les Problèmes d' ordonnancement avec contraintes dijonctives*. Note DS 9 bis, SEMA, Montrouge, Paris.

Sabuncuoglu, I. e Bayiz, M. (1997). *A Beam Search Based Algorithm for the Job Shop Scheduling Problem*. Research Report: IEOR–9705, Department of Industrial Engineering, Faculty of Engineering, Bilkent University, Ancara, Turkey.

Sampson, S.E. e Weiss, E.N. (1993). Local Search Techniques for the Generalized Resource Constrained Project Scheduling Problem. *Naval Research Logistics*, Vol. 40, pp. 665–675.

Schaffer, J.D., Caruana, R.A., Eshelman, L.J. e DAS, R. (1989). A Study of Control Parameters Affecting Online Performance of Genetic Algorithm for Function Optimization. *Proceedings of the 3rd International Conference in Genetic Algorithms*.

Schaffer, J.D. e Grefenstette, J.J. (1985). Multi-Objetive Learning via Genetic Algorithm. *Proceeding of the 9th International Conference on Artificial Intelligence*, Los Angeles, pp. 593–595.

Schirmer, A. (1998). Case-Based Reasoning and Improved Adaptative Search for Project Scheduling. *Manuskripte aus den Instituten fur Betriebswirtschaftslehre*, N.º 472, University of Kiel, Germany.

Schirmer, A. e Riesenberg, S. (1997). Parameterized Heuristics for Project Scheduling – Biased Random Sampling Methods. *Manuskripte aus den Instituten fur Betriebswirtschaftslehre*, N.º 456, University of Kiel, Germany.

Schrage, L. (1970). Solving Resource-Constrained Network Proms by Implicit Enumeration-non-preemptive Case, *Operations Research*, Vol. 18 (2), pp. 225–235.

Schroeder, G.R. (1989). *Operations Management*. McGraw-Hill International Editions - Management Series.

Schwindt, C. (1998). *A branch-and-bound algorithm for the resource-constrained project duration problem subject to temporal constraints*. Technical Report 544, WIOR, Universitat at Karlsruhe, Karlsruhe, Germany.

Shankar, V. e Nagi, R. (1996). A flexible optimization approach to multi-resource, multi-project planning and scheduling. *Proceedings of 5th Industrial Engineering Research Conference*, Minneapolis, USA.

Sidney, J. B. (1973). An Extension of Moore's Due Date Algorithm. In: Elmaghraby, S.R. (ed.). *Symposium on the Theory of. Scheduling and its Application*, pp. 393–398.

Sidney, J.B. (1977). Optimal Single-Machine Scheduling with Earliness and Tardiness Penalties. *Operations Research*, Vol. 25, N.º 1, pp.62–69.

Slowinski, R., Soniewicki, B. e Weglarz, J. (1994). DSS for multiobjective project scheduling. *European Journal of Operational Research*, Vol. 79, pp. 220–229.

Smeds, P.A. (1980). Methods for checking the consistency of precedence constraints. *AIIE Transactions*, Vol. 12, pp. 171–178.

Smith, S.F. (1983). Flexible Learning of Problem Solving Heuristics through Adaptative Search. *Proceeding of the 8th International Conference on Artificial Intelligence*, pp. 422–425.

Smith, W.E. (1956). Various optimizers for single state production. *Naval Research Quarterly*, Vol. 3, pp.59–66.

Spears, W.M. e Dejong, K.A. (1991). On the Virtues of Parameterized Uniform Crossover. *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 230–236.

Speranza, M.G. e Vercellis, C. (1993). Hierarchical Models for Multi-project Planning and Scheduling. *European Journal of Operational Research*, Vol. 64, pp.312–325.

Sprecher, A. (1997). *Solving the RCPSP efficiently at modest memory requirements.* Working paper, University of Kiel.

Sprecher, A., Kolisch, R., e Drexl. A. (1995). Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *European Journal of Operational Research*, Vol. 80, pp. 94–102.

Stinson, J.P., Davis, E.W. e Khumawala, B.M. (1978). Multiple resource-constrained scheduling using branch and bound. *AIIE Transactions*, Vol. 10, pp. 252–259.

Storer, R.H., Wu, S.D. e Park, I. (1992). Genetic Algorithms in Problem Space for Sequencing Problems. *Proceedings of a Joint US-German Conference on Operations Research in Production Planning and Control*, pp. 584–597.

Storer, H.R., Wu, S.W. e Vaccari, R. (1992). *New Search Spaces for Sequencing Problems with Application to Job Shop Scheduling. Management Science*, Vol. 38, N.º 10, p. 1495–1509.

Storer, R.H., Wu, S.D. e Vaccari, R. (1995). Problem and Heuristic Space Search Strategies for Job Shop Scheduling. *ORSA Journal on Computing*, Vol. 7(4), pp. 453–467.

Szwarc, W. (1977). Optimal Two-Machine Orderings in the 3 x n Flow Shop Problem. *Operations Research*, Vol. 25, N.º 1, pp.70–77.

Taillard, Eric D. (1994). Parallel Taboo Search Techniques for the Job Shop Scheduling Problem. *ORSA Journal on Computing*, Vol. 6, N.º 2, pp. 108–117.

Talbot, B. e Patterson, J.H. (1978). An efficient integer programming algorithm with network cuts for solving resource constrained scheduling problems. *Management Science*, Vol. 24, pp. 1163–1174.

Tavares, L.V. (1986). Multicriteria scheduling of a railway renewal program. *European Journal of Operational Research*, Vol. 25, pp. 395–405.

Tersine, R. (1987). Production Operations Management. North Holland.

Thesen, A. (1976). Heuristic Scheduling of Activities Under Resource and Precedence Restrictions. *Management Science*, Vol. 23, pp. 412-422.

Thomas, P.R. e Salhi., S. (1997). An investigation into the relationship of heuristic performance with network-resource characterisitcs. *Journal of the Operational Research Society*, Vol. 48(1), pp. 34–43.

Thomas, P.R. e Salhi., S. (1998). A Tabu Search Approach for the Resource Constrained Project Scheduling Problem. *Journal of Heuristics*, Vol. 4, pp. 123–139.

Tormos, P. e Lova. A (2001). A Competitive Heuristic Solution Technique for Resource-Constrained Project Scheduling. *Annals of Operations Research*, Vol. 102, pp. 65–81.

Tsubakitani, S. e Deckro, R.F. (1990). A Heuristic for Multi-Project Scheduling with Limited Resources in the Housing Industry. *European Journal of Operational Research*, Vol. 49. pp. 80–91.

Ulusoy, G. e Ozdamar, L. (1989). Heuristic Performance and Network/Resource Characteristics in Resource-constrained Project Scheduling, *Journal of the Operational Research Society*, Vol. 40, pp. 1145–1152.

Vaca, L. (1995). *Um Algoritmo Evolutivo para a Programação de Projectos Multi-Modos com Nivelamento de Recursos.* Tese de Doutorado, Florianópolis.

Vaessens, R.J.M., Aarts, E.H.L. e Lenstra, J.K., (1996). Job Shop Scheduling by local search. *INFORMS Journal.*

Valls, V., Perez, M.A. e Quintanilha, M.S. (1992). *Heuristic performance in large resource-constrained projects, Working Paper, Department D'Estadistica I Investigacio Operativa.* Universitat de Valencia, Spain.

VanDerMerwe, A.P. (1997) Multi-project management – organizational structure and control. *International Journal of Project Management*, Vol. 15, N.º 4, Elsevier Science, Oxford, Engl IS: 0263–7863, pp. 223–233.

Vercellis, C. (1994) Constrained multi-project planning problems: A Lagrangean decomposition approach. *European Journal of Operational Research*, Vol. 78, pp. 267–275.

Venugopal, V. e Naredran, T.T. (1992). A Genetic Algorithm Approach to the Machine-component Grouping Problem With Multiple Objectives. *Computers and Industrial Engineering*, Vol. 22 (4), pp. 469–480.

Villeneuve, L., Gharbi, A., e Pellerin, R. (1997). *Nouvelle approche d'ordonnancement multi-project pour l'industrie de la refection.* XII Congrés International Franco-Quebecois de Génie Industriel ALBI.

Wagner, H.M. (1959). An integer programming model for machine scheduling. *Naval Research Logistics Quarterly*, Vol. 6, pp. 131–140.

Wang, L. e Zheng, D. (2001). An effective hybrid optimisation strategy for job-shop

scheduling problems. *Computers & Operations Research*, Vol. 28, pp. 585–596.

Whitehouse, G.E. e Brown, J.R. (1976). Genres: An Extension of Brooks Algorithm for Project Scheduling with Resource Constraints. *Computers & Industrial Engineering*, Vol. 3, pp. 261–268.

Whitley, D., Starkwether, T. e Shaner, D. (1991). The Travelling Salesman and Sequence Scheduling Problems: Quality Solutions Using Genetic Edge Recombinations. In: Davis, L. *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, pp. 350–372.

Wiley, V.D., Deckro, R.F. e Jackson, J.A, (1998). Optimization analysis for design and planning of multi-project programs. *European Journal of Operational Research*, Vol. 107, pp. 492–506.

Williamson, D.P., Hall, L.A., Hoogeveen, J.A., Hurkens, C.A.J., Lenstra, J.K., Sevast'janov, S.V. e Shmoys, D.B. (1997). Short Shop Schedules. *Operations Research*, March - April, Vol. 45(2), pp. 288–294.

Yang, K.K., Talbot, F.B. e Patterson, J.H. (1992). Scheduling a project to maximize its net present value: na integer programming approach. *European Journal of Operational Research*, Vol. 64, pp. 188–198.

Zamani, M.R. (2001). A high-performance exact method for the resource-constrained project scheduling problem. *Computers & Operations Research*, Vol. 28, pp. 1387–1401.

Zhou, H., Feng, Y. e Han, L. (2001). The hybrid heuristic genetic algorithm for job shop scheduling. *Computers & Industrial Engineering*, Vol. 40, pp. 191-200.

POLITÉCNICO DO PORTO

POLITEMA

JORGE MAGALHÃES MENDES

PLANEAMENTO DE PROJECTOS COM RECURSOS LIMITADOS

FIPP