Universidade do Minho
Escola de Engenharia

José Filipe Moreira da Silva Figueiredo

Implementação de um Algoritmo Genético Híbrido com Simulated Annealing para o problema Job Shop

janeiro de 2015

José Filipe Moreira da Silva Figueiredo **Implementação de um Algoritmo Genético Híbrido com Simulated Annealing para o problema Job Shop**

UMinho | 2015

Universidade do Minho
Escola de Engenharia

José Filipe Moreira da Silva Figueiredo

Implementação de um Algoritmo Genético Híbrido com Simulated Annealing para o problema Job Shop

Dissertação de Mestrado
Mestrado em Engenharia de Sistemas

Trabalho efetuado sob a orientação do
Professor Doutor José António Vasconcelos Oliveira

janeiro de 2015

DECLARAÇÃO

Nome:
José Filipe Moreira da Silva Figueiredo

Endereço electrónico: josefilipefig@gmail.com

Número de Bilhete de Identidade: 12697413

Título dissertação:
Implementação de um algoritmo genético híbrido com simulated annealing para o problema job shop.

Orientador:
Professor Doutor José António Vasconcelos Oliveira.

Ano de Conclusão: 2015

Designação do Mestrado:
Mestrado em Engenharia de Sistemas

Universidade do Minho, / /

Assinatura:

Agradecimentos

Este espaço é dedicado àqueles que, de alguma forma, contribuíram para que esta dissertação fosse realizada. As palavras que aqui deixo são sentidas e sinceras.

Gostaria de agradecer ao meu orientador Professor Doutor José António Vasconcelos Oliveira, pelo apoio, disponibilidade e confiança depositada em mim, bem como no acompanhamento e supervisão deste trabalho. A sua orientação foi fundamental para a conclusão deste estudo.

Aos meus colegas do Mestrado em Engenharia de Sistemas pelo apoio e partilha de conhecimento que seguramente enriqueceram a minha formação académica.

Um especial agradecimento aos dois membros do meu grupo de trabalho, André Carvalho e Ivo Macedo cujo empenho e dedicação muito contribuíram para este projeto.

Uma palavra de agradecimento à Fundação para a Ciência e Tecnologia e ao "Programa Operacional Fatores de Competitividade – COMPETE" por me ter proporcionado a participação numa conferência internacional com a publicação de um artigo científico.

À Universidade do Minho por todos os recursos que disponibiliza para um desenvolvimento pessoal e académico de excelência.

À Rita, um agradecimento especial pelo apoio e carinho diários. Pela transmissão de confiança e força, em todos os momentos. Por tudo, a minha enorme gratidão.

À minha Família, em especial aos meus Pais e Irmão, um enorme obrigado por acreditarem sempre em mim e por todo o carinho, confiança e ensinamentos que ao longo de toda a vida me transmitiram. Dedico-vos todo este trabalho.

O término desta etapa será o começo de uma nova.

Resumo

Neste trabalho apresenta-se um estudo sobre problemas de planeamento de operações do tipo job shop. Devido à sua natureza combinatória, o planeamento de operações deste tipo pertence à classe de problemas *NP-difícil*. Dada a complexidade destes problemas, torna-se impraticável testar todas as soluções possíveis pois tal não seria exequível em tempo computacional útil. Mesmo com a utilização de métodos de solução exata aplicados a modelos de programação inteira a solução do problema fica limitada a instâncias de pequena dimensão. Por este motivo, vai-se ao encontro do paradigma de investigação que tem sido desenvolvido nas últimas duas décadas que procura encontrar soluções recorrendo a métodos de solução aproximada.

Desde a pesquisa por arrefecimento simulado (Simulated Annealing) até à optimização por enxame de partículas (Particle Swarm Optimization) é possível ainda encontrarem-se muitas variantes dentro da mesma classe de métodos aproximados.

Um dos métodos que se tornou muito popular na comunidade científica é o Algoritmo Genético. A simplicidade desta técnica na representação de modelos complexos e também a facilidade de integração com outros métodos de resolução são os fatores que justificam o seu estudo. Contudo é necessário manter uma diversidade genética ao longo das iterações para evitar uma convergência prematura para mínimos locais. Para colmatar esta ineficiência recorre-se à sua hibridização, combinando com outro método de aproximação (Simulated Annealing).

A abordagem proposta nesta dissertação pretende dar mais um contributo para a resolução do problema de planeamento de operações em ambiente do tipo job shop recorrendo a um algoritmo genético híbrido com arrefecimento simulado (Simulated Annealing). É realizada uma caracterização das tarefas e recursos do problema Job Shop bem como um estudo de Algoritmos Genéticos e seus componentes.

Como fator diferenciador em relação aos trabalhos já publicados, pretende-se desenvolver um método que permita resolver o problema job-shop clássico, bem como a hibridização de métodos de solução aproximada.

Os métodos implementados são validados através da análise dos resultados obtidos para aferir a sua eficácia e eficiência. Os testes realizados têm como base conjuntos de problemas padronizados, previamente definidos por autores reconhecidos na área.

Abstract

This paper presents a study of job shop scheduling problems (JSSP). Due to its combinatorial nature, JSSP belongs to NP-hard class problems. Given the complexity of such problems, it becomes impossible to test all possible solutions in order to find the optimal solution of the problem, as this would not be feasible in useful computational time. Even with the use of exact solution methods applied to integer programming models, it is limited to small instances. For this reason, it will be followed the research paradigm that has been developed over the past two decades to find solutions using approximate methods.

Since the research by simulated annealing to the particle swarm optimization it is possible to find many variants within the same class of approximate methods.

One method that has become very popular in the scientific community is the Genetic Algorithm. The simplicity of this technique in the representation of complex models and also the ease of integration with other methods of resolution are the factors that justify their study. However genetic algorithms are not effective in finding the optimal solution value, but the regions where the solution is. To transcend this inefficiency one solution is the hybridization, combining with another approximation method (Simulated Annealing), whose goal is to find the optimal solution in a limited region.

The approach proposed in this thesis aims to further contributes to solving the problem of planning of operations in job shop type environment using a hybrid genetic algorithm with simulated annealing. It is made a characterization of tasks and resources of the JSSP and a study of Genetic Algorithms and their components.

As a differentiating factor in relation to the work already published, we intend to develop a method for solving the classical job shop problem, as well as the hybridization of distinct approximation methods.

The implemented methods are validated through the analysis of the results to check their effectiveness and efficiency. The tests are based on standardized sets of problems, as established by recognized authors in the area.

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

TABELA DE CONFIGURAÇÕES

1. Introdução

1.1- Enquadramento

Ao longo dos últimos anos, a competitividade do sistema produtivo tem vindo a aumentar exponencialmente. Este aumento pode ser explicado pela abertura ao mercado global. Por um lado, possibilitou o acesso a novos mercados com técnicas e recursos inovadores, mas por outro lado, aumentou a concorrência a que as organizações estão sujeitas. Como consequência, tornou-se fundamental uma reavaliação dos processos de produção das organizações, de forma a garantir uma utilização eficiente e eficaz dos recursos disponíveis e assim aumentar a sua capacidade de resposta a novas ameaças externas. Este aumento de competitividade global faz paralelo com o aumento da exigência dos clientes, o que obriga a um aumento de produtividade e qualidade por parte das organizações.

Este novo paradigma de concorrência dos mercados, levou a um reforço da importância do planeamento e sequenciamento das operações produtivas. O planeamento tem como base os objetivos que se pretendem atingir, a definição das atividades nucleares da organização na definição dos recursos disponíveis, nas restrições tecnológicas, físicas, organizacionais e de dependência entre as atividades que constituem o processo produtivo das organizações. O sequenciamento visa estabelecer a ordem ou a sequência das atividades estabelecidas na fase de planeamento. É nesta fase que se calcula o tempo de início e fim de cada atividade e o resultado final será a geração de um plano base de sequenciamento das atividades.

O problema job shop estudado neste trabalho é um problema de optimização combinatória, de elevada complexidade (NP-Difícil) (**Figura 1**). A abordagem proposta nesta dissertação segue a modelação clássica.

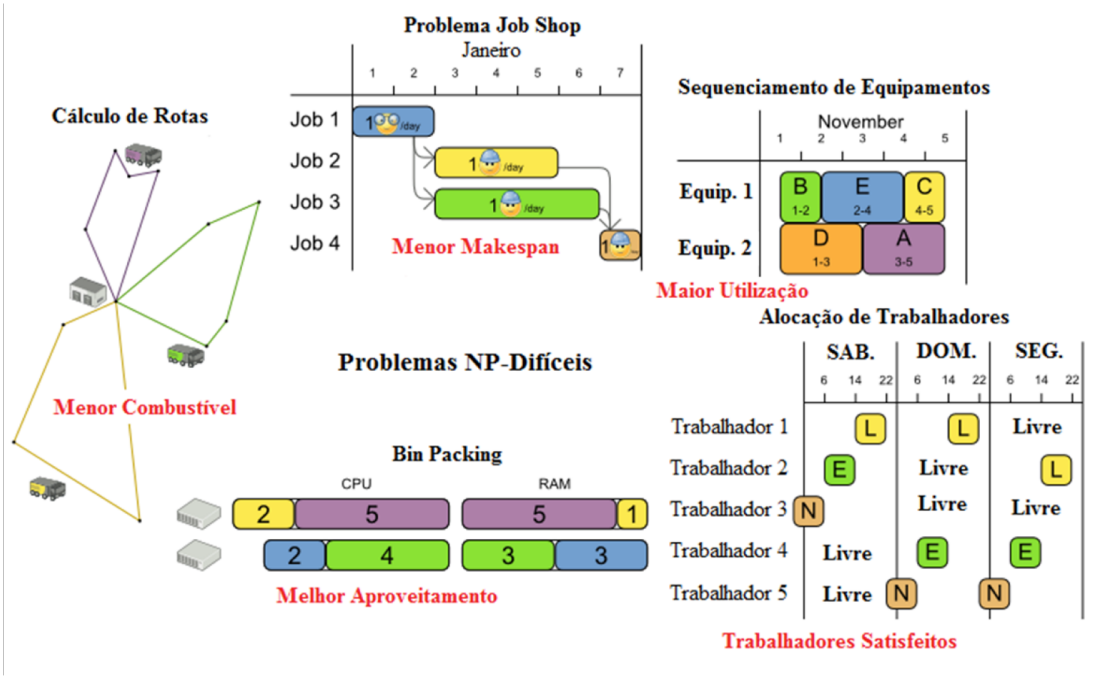


Figura 1 - Problemas NP-Difíceis
Fonte: Adaptado de http://www.optaplanner.org/

1.2- Objetivo da Dissertação

A prática de planeamento e sequenciamento de atividades (ou operações) proporciona benefícios significativos para as organizações e pessoas. A preocupação crescente na utilização eficaz e eficiente dos recursos disponíveis, cria uma série de vantagens competitivas face ao mercado cada vez mais exigente. Os benefícios são vários entre os quais se destaca a diminuição dos custos através da redução de gastos desnecessários, diminuição do tempo de conclusão das atividades, aumento da produtividade e qualidade do produto ou serviço e racionalização dos recursos usados.

O tema de investigação deste trabalho centra-se no problema de planeamento/sequenciamento de operações em ambiente do tipo job shop. Pretende-se fazer uma caracterização das tarefas e recursos subjacentes ao problema e às várias alternativas para a sua resolução.

O método de resolução que foi selecionado para resolver o problema é o algoritmo genético que, pela sua simplicidade na representação de modelos complexos e também a facilidade de integração com outros métodos de resolução, justificam a sua escolha.

Tendo em vista uma convergência e melhoria dos resultados do algoritmo genético, pretende-se combinar com outros métodos nomeadamente o arrefecimento simulado (Simulated Annealing).

Com este trabalho de investigação pretende-se desenvolver um sistema de apoio à decisão que implemente os métodos propostos e gere uma solução, que podendo não ser óptima, seja de boa qualidade.

1.3- Motivação

Nos tempos de austeridade em que vivemos, torna-se fundamental a eficiente utilização dos recursos existentes. São conhecidos os maus exemplos de gestão que levaram à falência de empresas outrora rentáveis. A concorrência agressiva dos mercados, a procura de maior produtividade com recursos mais limitados e a satisfação do cliente final implicam um novo paradigma de gestão de recursos. É necessário decidir melhor e mais rápido que a concorrência, utilizando apenas os recursos disponíveis para a conclusão do produto final. Para isto é necessário um planeamento rigoroso de todas as etapas da produção.

Os problemas de otimização e planeamento de operações estão presentes no nosso dia-a-dia. Como planear as rotas dos camiões de recolha de resíduos urbanos, determinar a melhor ordem de entrega de correio, planificar horários das disciplinas a alunos ou como alocar trabalho aos funcionários no espaço fabril, são problemas reais e complexos que requerem uma análise intensiva e funcional.

Ao longo do seu percurso universitário, o autor foi confrontado com diversos problemas de otimização que lhe despertaram o gosto pela análise, modelação e resolução desses sistemas complexos. Mais especificamente o problema job shop, que não sendo um problema recente, é um problema que tem sido alvo de vários estudos devido à sua importância no espaço fabril. A evolução que tem sofrido deve-se ao fato de se aplicar a novas abordagens e técnicas de resolução, bem como os avanços tecnológicos que se tem alcançado.

A principal motivação do autor insere-se na procura de métodos computacionalmente viáveis para a resolução deste tipo de problemas que consigam encontrar soluções de boa qualidade e em tempo computacional útil. Os métodos

recomendados pela bibliografia existente são os métodos aproximados de otimização. Com esta dissertação procura-se desenvolver e implementar alguns destes métodos para obter resultados válidos no problema job shop e que permitam tirar conclusões sobre este tipo de técnicas.

1.4- Questões de Investigação

As questões de investigação a que se pretende dar resposta com este trabalho centram-se nos métodos de resolução a aplicar para o problema job shop e os benefícios da sua hibridização. Com instâncias de teste de solução conhecida, pretende-se avaliar o desempenho do método utilizado.

Questões pertinentes para o problema:

- Pode a integração do Simulated Annealing no Algoritmo Genético melhorar os resultados obtidos?
- Será o Algoritmo Genético o melhor método para a resolução do job shop clássico?

Para instâncias de teste de solução conhecida levantam-se as seguintes questões:

- Quantas iterações do AG são necessárias para encontrar uma solução de boa qualidade?
- Qual o tamanho ideal da população?
- Qual a taxa de cruzamento padrão, para uma evolução constante do AG?

1.5- Metodologia e Revisão da Literatura

O processo de revisão crítica da literatura constitui os alicerces em que se fundamenta a investigação a desenvolver. Para a elaboração desta dissertação foram estudadas as teorias chave que dão base aos problemas de planeamento de operações, bem como foi realizado um levantamento de diferentes fontes bibliográficas. O tema mostra-se atual e vai ao encontro com investigação previamente publicada.

Para o planeamento da pesquisa bibliográfica inicialmente foram definidos os parâmetros, palavras-chave e termos de pesquisa. Em seguida definiram-se as bases de dados a pesquisar, como a biblioteca de conhecimento *online B-On* e a biblioteca geral da Universidade do Minho.

Tendo como base o planeamento efetuado, recorreu-se a fontes bibliográficas terciárias, tais como índices e resumos para tentar reunir informação relevante para o tema. Nesta fase, localizaram-se artigos importantes referenciados em livros e artigos de revistas científicas.

A maior fonte de conhecimento para a dissertação teve como base fontes primárias e secundárias tais como relatórios e teses académicas, livros e artigos sobre o tema de investigação, pesquisas na internet e a preciosa ajuda do orientador da dissertação.

O processo de revisão de literatura foi contínuo de modo a garantir a atualidade da informação e sobretudo garantir que todo o material pertinente é revisto.

1.6- Estrutura da Dissertação

Neste trabalho apresenta-se um estudo sobre problemas de planeamento de operações em ambiente do tipo job shop e está divido em sete capítulos distintos. Seguidamente descrevem-se os conteúdos abordados em cada capítulo. Procurou-se que a transição entre capítulos seguisse uma lógica clara e sistemática de forma a tornar uma leitura estruturada e fluente.

No **Capítulo 1** formula-se e clarifica-se o tópico de investigação. É realizada uma introdução ao tema, definindo-se a motivação e os objetivos que se pretendem atingir. Apresenta-se o planeamento da pesquisa bibliográfica e da metodologia a utilizar. Este é o ponto de partida para se atingir os objetivos propostos.

No **Capítulo 2** apresenta-se a revisão de literatura. Clarifica-se o estado da arte nas últimas três décadas, e qual a tendência de evolução do tema. Faz-se uma revisão acerca do problema de planeamento de operações em ambiente job shop, dos métodos de otimização e pesquisa de soluções exatas e aproximadas e à sua representação gráfica. Apresentam-se as opções tomadas para a resolução do problema, bem como uma análise detalhada de algoritmos genéticos e dos seus componentes. Adicionalmente, é apresentada uma revisão do algoritmo de

arrefecimento simulado, que é utilizado para efeito de melhoria e convergência das soluções encontradas.

No **Capítulo 3** apresenta-se a metodologia usada para a geração de soluções admissíveis. Faz-se a descrição dos pontos fortes e fracos dos algoritmos genéticos e quais as alternativas de resolução. Faz-se uma referência aos tipos de representação, operadores genéticos, parâmetros e robustez destes métodos, bem como a sua hibridização com algoritmos de arrefecimento simulado.

No **Capítulo 4** inicia-se a abordagem proposta para o problema de planeamento de operações em ambiente do tipo job shop. Inicia-se com a descrição e formulação do problema. Apresentam-se as decisões tomadas e os métodos de representação gráfica usados. Faz-se uma definição das máquinas, recursos, restrições presentes e da metodologia de geração de planos.

No **Capítulo 5** descrevem-se as experiências computacionais realizadas para os pressupostos deste estudo e avaliam-se os resultados encontrados. Faz-se uma comparação fundamentada com outros sistemas de geração de soluções para o problema job shop.

No **Capítulo 6** apesentam-se as conclusões gerais do trabalho e descrevem-se algumas perspetivas de trabalho futuro.

2. Revisão da Literatura

2.1- Introdução

O conceito de planeamento não é novo. Os antigos egípcios construíram as pirâmides há cerca de 4500 anos. Enormes ferrovias transcontinentais foram construídas durante o séc. XIX e edifícios de diferentes tamanhos e complexidade foram construídos há tanto tempo quanto o Homem se tornou sedentário. Estas grandes obras implicaram certamente planificação dos trabalhos a realizar. Todavia apenas na segunda metade do séc. XX se começou a falar formalmente sobre planeamento de projetos.

Nas próximas secções é abordado a área de sequenciamento. Embora fortemente ligados, Baker e Su (1974) diferenciam os conceitos de planeamento e sequenciamento. Que produto ou serviço produzir? Qual a quantidade a produzir? Que recursos usar? Estas perguntas pertencem à função de planeamento. Em contraste, a função de sequenciamento assume que essas respostas já existem. No planeamento identificam-se as operações a fazer, e determina-se a quantidade de recursos a utilizar. O sequenciamento apenas se preocupa em como afetar os recursos disponíveis para completar as operações definidas.

Em seguida faz-se uma revisão de problemas de sequenciamento em ambiente do tipo job shop e dos seus métodos de resolução. Brucker et al. (1992) considera este problema como um dos mais difíceis problemas de sequenciamento, e teve a sua origem há mais de 5 décadas, de acordo com Fisher e Thompson (1961).

2.2- Representação Gráfica

Para a representação gráfica de problemas de sequenciamento é de salientar os mapas de Gantt e os grafos disjuntivos. Em seguida analisam-se estes modelos e explica-se a sua origem.

2.2.1- Mapa de Gantt

De acordo com Field e Keller (1998), Meredith e Mantel (1995) e Nicholas (1990) o conceito de sequenciamento obteve importantes desenvolvimentos com o trabalho de Gantt durante a I Guerra Mundial.

H. Gantt (1916) discute o sequenciamento de operações, especialmente em ambiente job shop, e define os diagramas de barras que mais tarde ficariam conhecidos como mapa de Gantt (Clark e Gantt, 1922). Em 1911 Taylor descreveu uma versão inicial dos diagramas num artigo intitulado "*Shop Management"*.

Segundo Porter (1929) e Alford (1945), estes diagramas foram revolucionários, pois relacionavam as atividades com o tempo num modo gráfico que permitia que se calculasse o horário de trabalho.

O mapa de Gantt caracteriza-se por ser de fácil leitura e de análise simples e clara. É um gráfico constituído por um sistema de eixos coordenados, representando no eixo das ordenadas as atividades e no eixo das abcissas o tempo. O tempo é normalmente definido em dias, semanas ou meses. Cada atividade é representada por um retângulo, no qual a largura corresponde à duração da atividade, e a altura corresponde à quantidade de recurso utilizada, por unidade de tempo.

Na **Figura 2** apresenta-se um exemplo de um mapa de Gantt aplicado a um problema de gestão de projetos. O projeto é constituído por seis atividades (1,2,3,4,5,6) e dois recursos (R1,R2) com capacidades de 4 e 2 respetivamente. A atividade 4 tem início no instante 4 e termina no instante 7. Durante o seu processamento gasta 2 unidades de R1 e 1 unidade de R2. Por outro lado, a atividade 3 apenas usa 2 unidades de R1 para o seu processamento e tem início no instante 1. A atividade 6 é a última a ser processada, determinando o tempo final do projeto no instante 13.

A principal vantagem da utilização destes mapas é a facilidade de planeamento que proporcionam. São diagramas bastante intuitivos orientados para o tempo, sendo fácil a adição ou remoção de novas atividades. Todavia apresentam algumas limitações, sobretudo no tratamento entre a dependência de atividades.

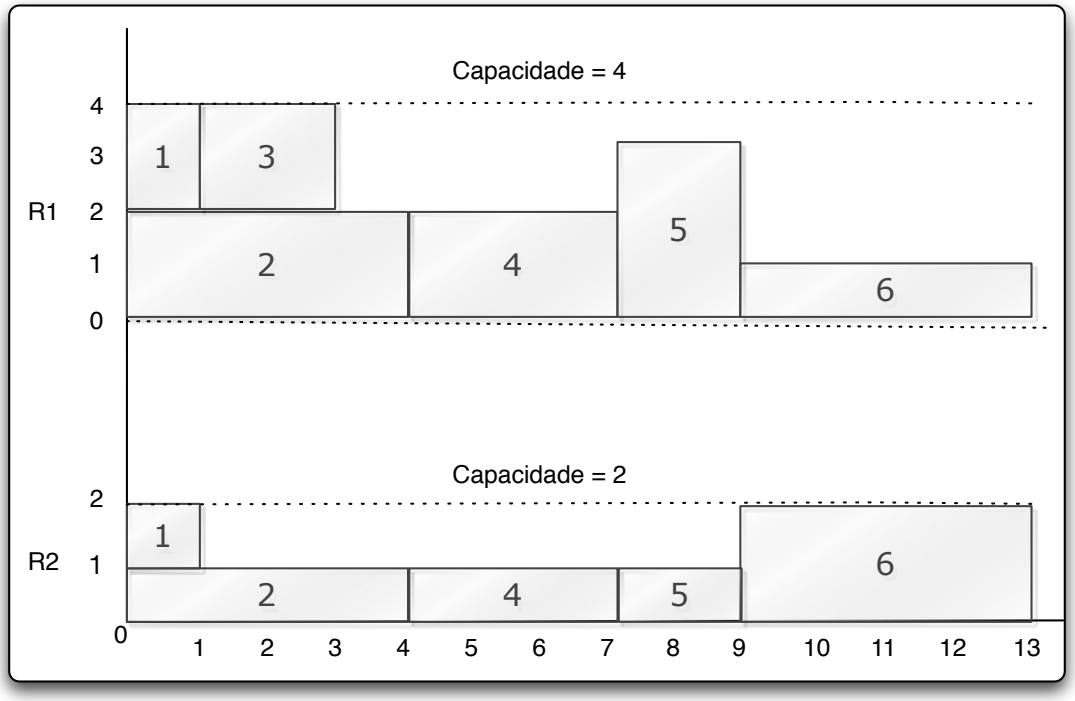


Figura 2 - Mapa de Gantt

2.2.2- Grafo Disjuntivo

O grafo disjuntivo proposto por Roy e Sussman (1964), é um dos modelos de representação gráfica mais populares para descrever instâncias de problemas de sequenciação do tipo job shop. Ao longo dos anos, foram propostas representações alternativas tal como Blazewicz et al. (2000), que representam um grafo disjuntivo como uma matriz de vizinhança, lista de sucessoras e predecessoras. Estas combinam vantagens da representação clássica com melhorias operacionais nos dados do problema. Em seguida formula-se a representação do modelo clássico.

O problema define-se como um conjunto de *n* tarefas (do termo inglês job), J_1, J_2, ..., J_n, e *m* máquinas dispostas numa oficina (do termo inglês *shop*). A cada tarefa J_i (ou ordem de fabrico) é associada a m_i operações, o_{i1}, o_{i2}, o_{im} com relações de precedência que definem uma ordem de processamento. Cada operação é processada numa única máquina durante um tempo fixo. Cada máquina apenas pode processar uma operação de cada vez sem interrupção.

O objetivo do problema consiste em afetar todas operações às máquinas que as processam, respeitando todas as restrições de capacidade e operacionais, no menor tempo possível. O instante de tempo da última operação sequenciada determina o

valor do *makespan*, que corresponde à duração da execução de todas as tarefas em menor espaço de tempo

O grafo disjuntivo é representado por G = (V, C U D). O conjunto dos vértices, V, é formado pelo conjunto de todas as operações o_1, o_2, ..., o_n. O conjunto V contem ainda duas operações fictícias (e de duração nula), o_0 e o_{n+1} que correspondem respetivamente, ao inicio e fim de todas as tarefas.

O conjunto dos arcos é formado por um conjunto C de arcos conjuntivos (arcos orientados), que representam as restrições de precedência entre as operações da mesma tarefa. Por cada arco pertencente ao conjunto C, existe um par ordenado de operações (*i* , *j*) relacionada por uma por uma precedência de *i* para *j*.

Para cada máquina (M_k) é definida um conjunto D_k de arcos disjuntivos (arcos não-orientados) que representam as operações (de tarefas diferentes) que requerem a mesma máquina para a sua execução. Cada máquina apenas pode processar uma operação de cada vez e por seu lado, cada operação, só pode ser processada numa única máquina. Cada arco toma um valor positivo igual ao tempo de processamento da tarefa associada.

Uma solução válida para o problema implica que o grafo resultante da orientação dos arcos disjuntivos é acíclico e a ser ótimo, o comprimento do caminho mais longo desde o vértice inicial até ao vértice final seja o menor. Este caminho mais longo determina o *makespan*,.

A **Tabela 1.** descreve um problema job shop com 3 máquinas M = {*M1, M2, M3*} e um conjunto de 3 ordens de fabrico J = {*J1 ,J2, J3*}, que descrevem a seguinte sequência de fabrico: J1: T1-> T2 -> T3 , J2: T4-> T5 , J3: T6-> T7 -> T8. Cada atividade Ti de duração Pi e máquina M(Ti).

Tabela 1 - Ordens e tempos de fabrico das operações

Ordens de fabrico	Operação 1		Operação 2		Operação 3	
	Máquina	Duração	Máquina	Duração	Máquina	Duração
1	1	20	2	10	3	5
2	3	5	2	20	-	-
3	2	12	1	4	3	18

O grafo disjuntivo da **Figura 3** contem toda a informação que é necessário para descrever uma solução parcial ou completa do problema de sequenciamento job shop. Na solução do problema é estabelecida uma ordem de processamento entre todas as operações que são processadas pela mesma máquina.

O grafo resultante da orientação dos arcos disjuntivos deve ser acíclico, de modo a garantir a precedência das operações. O comprimento do caminho mais longo, também designado por caminho crítico, entre o início e o fim da rede deve ser o menor. O valor do *makespan* é igual ao do caminho crítico. Qualquer arco (i, j) pertencente ao caminho crítico designa-se por arco crítico.

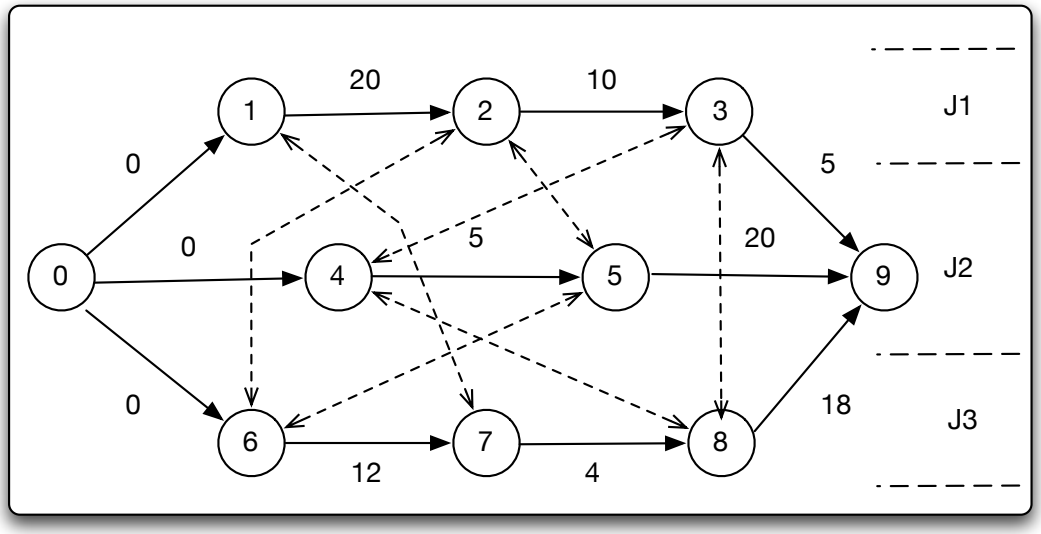


Figura 3 - Grafo Disjuntivo

Na **Figura 4** é representado o caminho crítico, relativo ao problema da **Tabela 1**. O caminho mais longo tem um valor total de 94 e passa pelos vértices 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9.

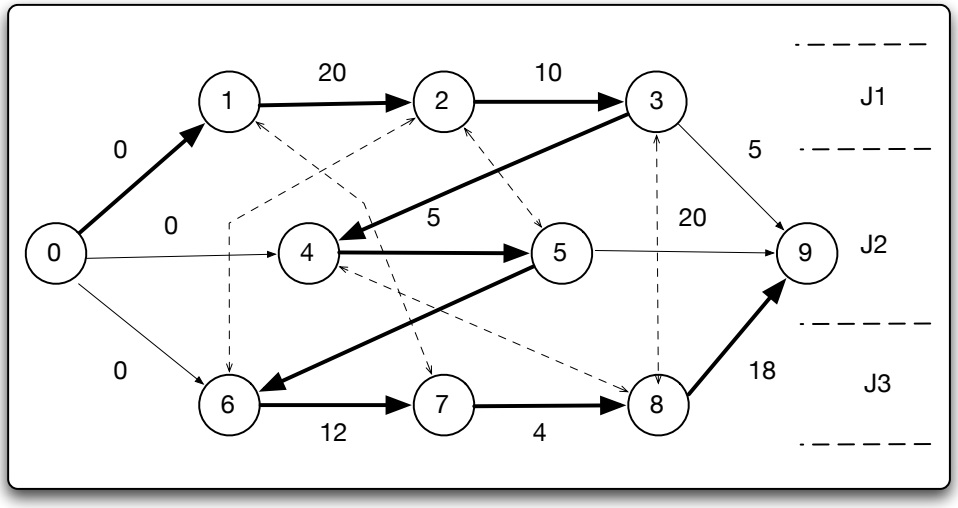


Figura 4 - Grafo Disjuntivo e Caminho Crítico

2.2.3- Tipos de sequenciamento

Para melhor compreensão dos problemas de sequenciamento, torna-se necessário a sua classificação. Antes demais, é necessário definir o que é um sequenciamento admissível. Baker e Su (1974) definem como sendo um sequenciamento de todas as atividades de um projeto que respeite os limites da capacidade dos recursos e de todas as restrições lógicas. O sequenciamento resultante diz-se ótimo se for admissível e se for tão bom ou melhor que qualquer outro plano admissível.

Em seguida apresenta-se um exemplo de um problema de sequenciamento com 2 ordens de fabrico a ser processadas em 2 máquinas para um melhor entendimento dos tipos de sequenciamento. Os dados do problema exemplo encontram-se na **Tabela 2**.

Tabela 2 - Problema de Sequenciamento

Ordens de fabrico	*Operação(Máquina; Tempo)*	*Operação(Máquina; Tempo)*
1	$O_{11}(2\ ;\ 4)$	$O_{12}(1\ ;\ 2)$
2	$O_{21}(1\ ;\ 1)$	$O_{22}(2\ ;\ 3)$

Os estudos de Conway et al. (1967), Baker e Su (1974), Kan (1976), French (1982), indicam que o espaço de soluções dos problemas de sequenciamento do tipo job shop incluem 3 tipos de planos:

- *Semi-ativo*: um sequenciamento admissível designa-se por semi-ativo se nenhumas das operações puder se processada mais cedo, sem alterar a ordem de processamento dos recursos (não existe nenhum tempo de inatividade desnecessário), **Figura 5.**

- *Ativo:* um sequenciamento admissível designa-se por ativo se nenhuma operação puder começar mais cedo sem atrasar as outras operações ou violar alguma restrição, **Figura 6**. Este conjunto de sequenciamento está incluído no conjunto dos semi-ativos, como referem Baker (1974) e Kolish (1995).

- *Não-atrasado*: um sequenciamento admissível designa-se por não-atrasado se nenhum recurso ficar inativo, podendo iniciar o processamento de alguma operação (**Figura 7).** Estes sequenciamentos estão incluídos no conjunto dos ativos, Baker (1974) e Kolish (1995).

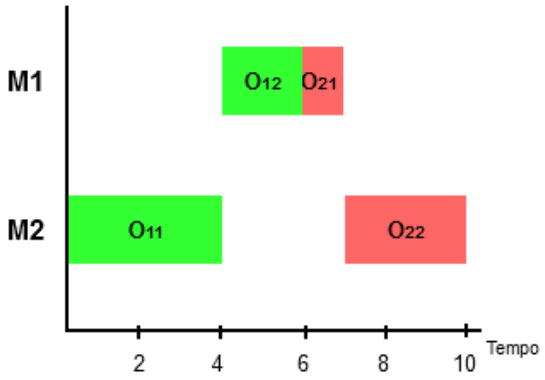


Figura 5 - Sequenciamento Semi-Ativo

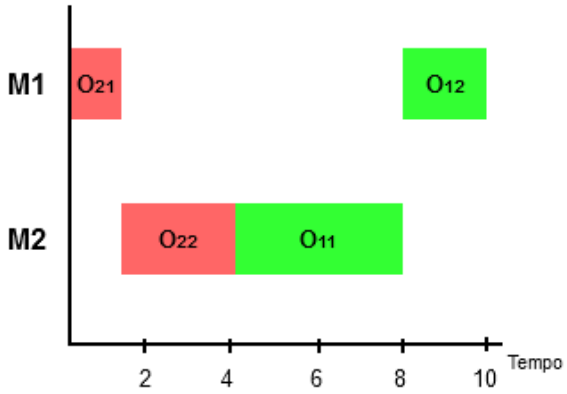


Figura 6 - Sequenciamento Ativo

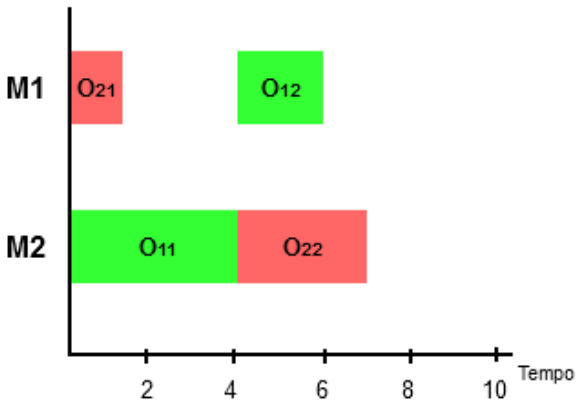


Figura 7 - Sequenciamento Não-Atrasado

Como se pode ver na **Figura 8**, a solução ótima pertence ao subconjunto dos planos ativos (Sprecher et al.,1995). Contudo em casos específicos a solução ótima pode ser um plano não-atrasado.

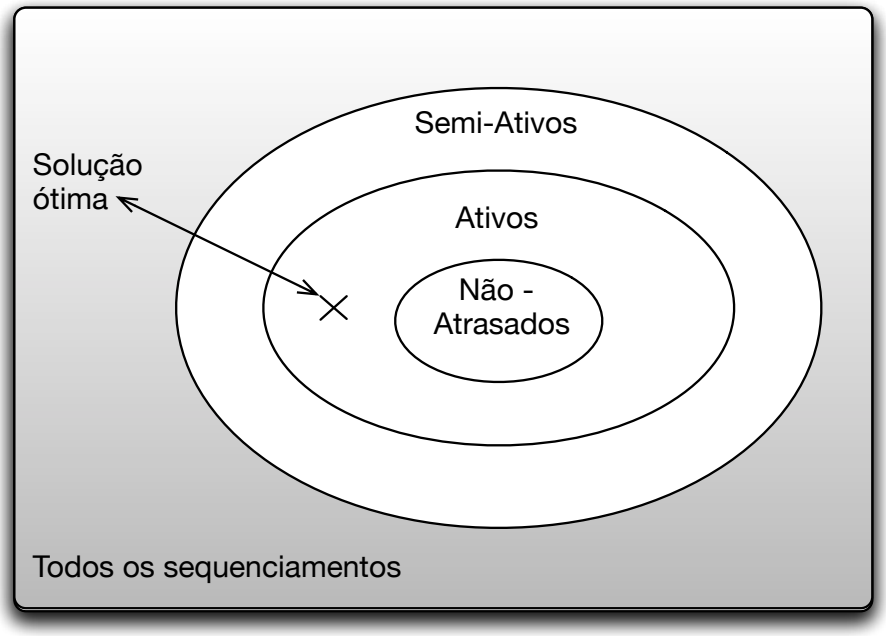


Figura 8 - Espaço de Soluções dos Sequenciamentos

2.3- Revisão do problema Job Shop

O problema de sequenciamento de operações job shop é um importante problema da optimização combinatória. Devido à sua complexidade é considerado um problema *NP-difícil* (Lenstra et al., 1977;, Kan, 1976; Garey e Johnson, 1979).

2.3.1- O modelo job shop clássico

Seja J = {1, ..., n} o conjunto de *n* tarefas (ou ordens de fabrico) a serem processadas no conjunto finito de *m* máquinas M = {1, ..., m}. Cada tarefa J_i é processada por todas as máquinas e é constituída por um conjunto de operações o_i = *{0, 1, ..., m}* que têm de ser sequenciadas numa ordem pré-estabelecida (*restrição de precedência*), interruptamente durante um instante de tempo fixo ρ_{oi}. Cada máquina apenas pode processar uma ordem (operação) de cada vez (*restrição de capacidade*) e cada ordem apenas pode ser processada por uma máquina de cada vez (*restrição disjuntiva*). O tempo final de sequenciamento de todas as operações de todas as ordens de fabrico refere-se como o makespan, $C_{max.}$

O modelo apresentado segue a notação usada por Blazewicz et. *al* (1996):

J_j - designa a tarefa j (ordem de fabrico) a ser processada.

o_i - designa a operação i . As operações são numeradas consecutivamente desde o_1 até o_N, em que N é o numero total de operações. No grafo disjuntivo as operações 0 e n+1 têm duração nula, e consideram-se operações fictícias pois representam o início e o fim de todas as ordens de fabrico.

M_k - designa a máquina *k* .

C_j - instante em que a ordem de fabrico J_j é concluída.

Um sequenciamento diz-se *admissível* se satisfaz todas as restrições. Um sequenciamento ótimo também satisfaz todas as restrições e é tão bom ou melhor que qualquer outro plano admissível.

2.3.2- Job shop com realimentação

No modelo clássico cada ordem de fabrico é constituída por um número de operações igual ao número de máquinas. Cada operação é processada numa máquina diferente, e duas operações sucessivas de J_i são processadas obrigatoriamente em máquinas diferentes.

No job shop com realimentação uma tarefa pode ser processada mais do que uma vez na mesma máquina. É possível também duas operações consecutivas serem processadas na mesma máquina.

Consequentemente o arco disjuntivo que une as operações da mesma ordem de fabrico a serem processadas na mesma máquina, deixa de fazer sentido e pode ser substituído pelos arcos conjuntivos existentes que definem as relações de precedência entre as operações da mesma tarefa.

2.4 – Métodos Exatos

Os métodos exatos têm um grande interesse teórico, pois permitem obter diferentes abordagens de métodos e regras. Todavia o seu interesse prático é pequeno, dado os pressupostos em que se baseiam, como refere Schroeder et al. (1989).

O problema de sequenciamento de n ordens de fabrico em m máquinas foi resolvido para m=1,2 e para valores arbitrários de n. Existem também algoritmos que resolvem o problema com 3 máquinas. Para problemas com m ≥ 4 ainda não foram desenvolvidos algoritmos eficientes.

2.4.1 - Formulações Matemáticas

É reconhecido por muitos investigadores que os problemas de sequenciamento podem ser resolvidos com técnicas de programação matemática. Uma das formas mais comuns de formulação matemática para o problema job shop é o de programação inteira linear de Manne (1960), que é revisto em seguida.

2.4.2 - Formulação por Programação Inteira

A formulação por programação inteira consiste na formulação de um modelo matemático constituído por um conjunto de restrições lineares e uma única função objetivo, mas com uma restrição adicional de algumas variáveis de decisão serem inteiras (y_{ipk}). Estas variáveis inteiras são binárias e implementam as restrições disjuntivas.

Adams et al. (1988) propuseram um modelo de programação inteira que se descreve em seguida:

Minimizar t_{n+1}

sujeito a :

$t_j - t_i \geq p_i$ $\quad \forall \{ o_i , o_j \} \in C$ *(Restrição conjuntiva)*

$t_j - t_i \geq p_i$ *ou* $t_i - t_j \geq p_j$ $\quad \forall \{o_i , o_j \} \in D_k, \ \forall M_k \in M$ *(Restrição disjuntiva)*

$t_i \geq 0$ $\quad \forall o_i \in O$ *(Menor inicio de processamento)*

onde:

t_i - representa o instante de tempo mais cedo em que pode ser iniciado o processamento da operação o_i.

Restrição conjuntiva - asseguram a não-sobreposição do processamento de operações da mesma tarefa e definem as relações de precedência que existe entre elas.

Restrições disjuntivas - asseguram as não-sobreposição de processamento entre operações que requerem a mesma máquina. Este tipo de restrição disjuntiva permite a troca na ordem de processamento de operações.

Mesmo com formulações mais compactas continua a ser necessário um grande numero de restrições (Manne 1960). Giffler and Thomson (1960) mencionam que a programação inteira não levou a métodos práticos de soluções, enquanto que

French (1982) não acredita que seja computacionalmente admissível a formulação de programação inteira a problemas de sequenciamento. Nemhauser e Wolsey (1988) e Blazewicz et al. (1991) dão ainda mais relevo a essas dificuldades e indicam que os modelos de programação matemática ainda não estão suficientemente desenvolvidos para os problemas de sequenciamento. Como resultado estas técnicas apenas são capazes de resolver pequenas instâncias, num período de tempo razoável.

Os resultados mais importantes utilizando formulação matemática foram obtidos utilizando métodos de relaxação Lagrangiana com os trabalhos de Fisher et al. (1983), Van De Velde (1991), Della Croce et al. (1995), Hoitomt et al. (1993) e de métodos de decomposição com os trabalhos de Ashour (1967), Applegate e Cook (1991), Chu et al. (1992) e Krüger et al. (1995). Nos métodos de relaxação Lagrangiana as restrições de capacidade e precedência são relaxadas utilizando Multiplicadores de Lagrange não negativos, com penalizações incorporadas na função objetivo. Na abordagem através dos métodos de decomposição o problema original é dividido em sub-problemas mais simples e de menor dimensão que depois são resolvidos individualmente.

Os resultados computacionais obtidos por estes métodos são de elevado custo enquanto que as soluções resultantes são normalmente de má qualidade, resultando num largo desvio da solução ótima. Mesmo combinando estas formulações matemáticas com outras técnicas, os resultados não se mostram satisfatórios (Fisher et al., 1983; Applegate e Cook, 1991).

Não é surpresa que as abordagens matemáticas se mostrem inadequadas para o problema job shop. Em seguida realiza-se uma revisão da abordagem enumerativa para o problema JSSP utilizando técnicas de branch and bound.

2.4.3 - Branch and Bound

Os algoritmos de *branch and bound* (partição e avaliação) tiveram origem na década de 60 com os trabalhos de Brooks e White (1965), Ignall e Schrage (1965), Lomnicki (1965), Brown e Lomnicki (1966) e Greenberg (1968). Todavia as primeiras técnicas de branch and bound aplicadas ao problema de sequenciamento de operações job shop foram desenvolvidas por Balas (1969). Este método aplicava o modelo do grafo disjuntivo e apenas considerava operações críticas.

As técnicas de *branch and bound* utilizam uma árvore dinâmica para a representação do espaço de soluções de todas as sequências admissíveis. Para realizar a pesquisa é aplicada uma partição e avaliação de uma sequência de nodos da árvore. A procura tem início no nodo superior (raiz da árvore) e termina quando o último nível da árvore (folha) for avaliado. Esta sequência de nodos resultante é considerada admissível. Cada nodo de um nível *p* da árvore de procura representa uma sequência parcial de *p* operações (Agin, 1966).

O processo de *partição* (*branch*) determina num nodo não selecionado, o próximo conjunto de possíveis nodos a partir do qual a busca pode progredir. As duas estratégias mais usuais de partição, são os métodos de geração de planos ativos (Generating Active Schedules, GAS) e métodos de resolução de conflitos essenciais (Settling Essential Conflicts, SEC) como referem Lageweg et al. (1977) e Barker e McMahon (1985). Os métodos GAS foram introduzidos por Giffler and Thompson (1960). Nestes métodos cada nodo representa um sequenciamento parcial onde o processo de partição determina o conjunto de operações a serem sequenciadas. Nos métodos SEC o processo de partição determina que operação o_i deve ser sequenciada antes de o_j ou vice versa. Barker e McMahon (1985) indicam que o SEC oferece maior flexibilidade, e em geral, encontra melhores soluções que o GAS.

O processo de *avaliação* (*bounding)* seleciona a operação escolhida para continuar a procura. Em seguida determina o limite inferior e o melhor limite superior alcançado, para o valor mínimo da função objetivo do nodo selecionado. Na maioria dos métodos *branch and bound*, o cálculo do limite superior inicial tem por base métodos heurísticos. Este cálculo é o primeiro passo a ser feito antes de se iniciar o processo de procura. Se em qualquer nodo o limite inferior ultrapassar o valor do melhor limite superior, então não é necessário continuar a procura nesse ramo pois o melhor limite superior não vai melhorar. Neste caso a procura recomeça (volta atrás) no nodo mais alto não visitado da árvore. O processo termina quando todos os nodos foram implicitamente ou explicitamente avaliados. O cálculo dos limites nos nodos é essencial para as técnicas branch and bound pois evitam a necessidade de se fazer cálculos para todo o espaço de soluções.

Apesar de ser um problema *NP-Difícil* (Lenstra et al. 1977; Kan, 1976; Garey e Johnson, 1979) vários tipos de métodos de avaliação (bounding) são descritos na literatura (Poots, 1980; Carlier, 1982). Para a resolução do problema job shop Akers

(1956), Brucker (1988) e Brucker e Jurisch (1993) determinam o cálculo do limite inferior dividindo o problema em subproblemas de menor dimensão, geralmente em instâncias de uma máquina.

Para complementar as técnicas de branch and bound é usual a implementação de regras de inferência ou proposições que definem a ordem para certas operações. Computacionalmente estas regras são importantes pois diminuem o espaço de procura e consequentemente, o tempo computacional (Pinson, 1995).

Com o desenvolvimento da tecnologia, foram alcançadas melhorias na performance dos métodos de branch and bound para o problema job shop. Mesmo estando limitados a instâncias de pequena dimensão, é necessário um conhecimento detalhado da instância do problema, bem como poderosas regras de inferência e métodos de seleção para reduzir o espaço de procura.

É consensual na comunidade científica, que os métodos exatos de procura não apresentam a robustez necessária para atacar os problemas job shop. As atenções voltaram-se para os métodos aproximados que, mesmo não garantindo a solução ótima, apresentam soluções de boa qualidade com tempo computacional aceitável. São portanto, mais adequados a instâncias de maiores dimensões.

2.5 - Métodos Aproximados

Devido ao problema job shop ser considerando NP-difícil (Lenstra et al., 1977; Kan, 1976; Garey e Johnson, 1979) os métodos exatos não são capazes de produzir soluções ótimas em tempo computacional útil. Por este motivo a comunidade científica focou-se na aplicação de métodos aproximados, que apresentam soluções de boa qualidade em tempo útil.

Na bibliografia existente consideram-se duas categorias principais de métodos aproximados:

- Métodos construtivos;
- Métodos de procura local ou meta-heurísticas.

Os métodos construtivos iniciam-se com um plano vazio e através de regras de prioridade, são adicionadas as atividades até ser obtido um plano final.

Os métodos de procura local começam com um plano gerado por um método construtivo. Em seguida através de alterações sucessivas ao plano inicial é gerado um plano de melhor qualidade, até que um ótimo local seja alcançado. Para evitar uma convergência prematura num ótimo local, várias meta-heurísticas foram desenvolvidas tais como o tabu search, simulated annealing e algoritmos genéticos (Arts et al., 1994).

Nas secções seguintes descrevem-se mais detalhadamente os principais algoritmos de aproximação.

2.6 - Métodos Construtivos

2.6.1 - Regras de Prioridade

Os métodos aproximados aplicados ao problema job shop foram inicialmente desenvolvidos com base em regras de prioridade. Devido à facilidade de implementação e de reduzida exigência computacional tornaram-se bastante populares na comunidade científica (Baker e Su, 1974; French, 1982; Morton e Pentico, 1993). Em cada iteração, todas as operações disponíveis para sequenciamento são atribuídas uma prioridade. A operação de maior prioridade é a escolhida a ser sequenciada (Kolish, 1995).

Os primeiros trabalhos sobre regras de prioridade foram desenvolvidos por Jackson (1955, 1957), Smith (1956), Rowe e Jackson (1956), Giffler e Thompson (1960) e Gere (1966). É de salientar que o algoritmo de Giffler e Thompson serve de base a todas as regras de prioridade e que se distingue pelo fato de gerar planos ativos.

A mais conhecida lista de heurísticas de sequenciamento é apresentada por Panwalker e Iskander (1977) onde 113 regras de prioridade são revistas e classificadas. Outros exemplos de conhecidas regras de prioridade são revistas por Blackstone et al. (1982), Haupt (1989), Bhaskaran e Pinedo (1991). Contudo uma conclusão comum a vários estudos, e originalmente estudada por Jeremiah et al. (1964), é que para a medida de desempenho do *makespan* nenhuma regra única de

prioridade é dominante, mas sim uma combinação de diferentes tipos de regras de prioridade (Gonçalves e Beirão, 1999). O mais recente estudo comparativo é feito por Chang et al. (1996) onde avalia o desempenho de 42 regras de prioridade utilizando um modelo de programação linear. A sua análise indica que regras relacionadas com o menor tempo de processamento têm melhor desempenho do que regras relacionadas com o maior tempo de processamento. Por exemplo o algoritmo de Viviers (1983) incorpora três níveis de classes de prioridade dentro das heurísticas de menor tempo de processamento. O método mais comum de melhoramento do desempenho da solução é a combinação probabilística de regras de prioridade individuais. Os primeiros exemplos desta estratégia são de Crowston et al. (1963) e Fisher e Thompson (1963). Lawrence (1984) compara o desempenho de dez regras de prioridade individuais com uma combinação aleatória dessas regras e mostra que o método combinado proporciona resultados muito superiores, mas com maior tempo de computação. Outros métodos mais sofisticados usados para controlar a escolha de qual regra a aplicar incluem um algoritmo genético (Dorndorf e Pesch, 1995) e de lógica difusa (Grabot e Geneste, 1994).

As regras de prioridade podem-se classificar quanto à sua variação ao longo do tempo (Gonçalves e Mendes, 1994) e quanto ao tipo de informação que incorporam (Day e Hottenstein, 1970):

- Estáticas - As prioridades não variam com o tempo.
- Dinâmicas - As prioridades variam com o tempo.
- Locais - Apenas usam a informação da ordem de fabrico a que pertence
- Globais - Incorporam informação de outras máquinas.

Em seguida apresentam-se algumas das regras de prioridade mais comuns.

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.

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 atual.

SLACK/NOP - "Slack per Number of Operations". Regra baseada na anterior, sendo a folga dividida pelo número de operações a processar.

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.

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.

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.

2.6.2 - Shifting Bottleneck (S.B.)

O método heurístico conhecido por *shifting bottleneck* (termo anglo-saxónico) foi desenvolvido por Adams et al. (1988). É um método iterativo que se fundamenta na decomposição do problema principal em sub-problemas de máquina única, onde são resolvidos otimamente, identificando-se as máquinas que são críticas ao problema. Em cada iteração determina-se as datas de disponibilidade e de entrega dos sub-problemas de modo a sequenciar as máquinas que formam o problema inicial.

2.6.2.1 - Representação do problema S.B.

É necessário representar o problema utilizando um grafo G e estabelecer as precedências das operações. Adiciona-se duas operações virtuais {0, N} representando o inicio e o fim. Cada nó do grafo representa uma operação. As operações estão ligadas por arestas que representam as relações de precedência entre operações. Cada operação tem um tempo de processamento.

2.6.2.2 – Passos do problema S.B.

Seja M o conjunto de máquinas do problema e M_0 conjunto de máquinas já sequênciandas.

- *Passo 1* – M= {0, … , N} ; M_0 = {}
- *Passo 2* – Identificar a máquina *Bottleneck*, com k ∈ M \ M_0 e sequencia-la.
  M_0 = {…, k}
- *Passo 3* – Ressequenciar as máquinas de M_0.

- *Passo 4* – Se $M_0 = M$ então parar. Senão voltar ao passo 2.

Sempre que uma máquina é sequenciada, adiciona-se a aresta correspondente ao grafo G'. Para que uma nova máquina seja sequenciada, deve-se ter em conta as arestas (precedências), já adicionadas anteriormente.

A máquina *Bottleneck* representa a máquina crítica que tem mais influência sobre o makespan do problema. Para cada máquina ainda não sequenciada, tenta-se minimizar o maior atraso sem interrupções desta máquina, utilizando-se apenas operações que sejam processadas por ela. Por cada iteração do algoritmo, deve-se identificar a máquina *Bottleneck* até não haver mais máquinas para sequenciar.

2.6.3 - Satisfação de Restrições

Estas técnicas são exemplos de métodos interativos de aproximação onde se aplicam muitas das regras e estratégias utilizadas nos métodos de branch and bound. Estas procuram a redução do tamanho do espaço de procura através da aplicação de restrições que limitam a ordem na qual as variáveis são selecionadas e a sequência de valores possíveis de atribuir a cada variável. Embora estas técnicas procurem obter planos admissíveis, têm dificuldade na representação de restrições e exigem elevado esforço computacional. Geralmente estes métodos convergem para mínimos locais.

Resultados pouco satisfatórios foram alcançados por Caseau e Laburthe (1995), Pesch e Tetzlaff (1996) e Nuijten e Le Pape (1998).

2.6.4 - Beam Search

Enquanto que as regras de prioridade apenas escolhem uma operação possível dentro do conjunto de conflitos, as técnicas de branch and bound avaliam todas as operações possíveis, implicitamente ou explicitamente. A técnica de beam search utilizada por Morton e Pentico (1993) combina estas duas abordagens avaliando uma série de melhores soluções em qualquer ponto de decisão. Esta abordagem é generalizada por Glover e Laguna (1997). Uma versão melhorada desta técnica foi desenvolvida por Sabuncuoglu e Bayiz (1997) onde utilizam um conjunto de regras de prioridade na escolha dos melhores descendentes.

2.6.5 - Algoritmo de Giffler e Thompson

O algoritmo de Giffler e Thomson (1960) permite gerar todos os planos ativos para o problema JSSP e aplica-se a *n* tarefas com *m* máquinas.

O algoritmo inicia-se pela seleção da operação com menor tempo de conclusão, o_j. Em seguida determina-se todas as operações que correm na mesma máquina (Mk) e que começam antes de o_j terminar. Estas operações são colocadas no conjunto de conflitos. Uma operação é então selecionada do conjunto de conflitos e é sequenciada o mais cedo possível. Este procedimento é repetido até que todas as operações sejam sequenciadas. As regras de prioridade definem o método de seleção das operações do conjunto de conflitos. A regra mais simples utiliza uma seleção aleatória de uma operação, do conjunto de conflitos.

Uma operação pode ser sequenciada se dentro da sua ordem de fabrico já tiverem sido sequenciadas as suas operações predecessoras. Apenas uma operação de cada vez é sequenciada. Caso existam n*m operações, o algoritmo terá n*m estágios.

Para o estágio t :

P_t - Plano parcial formado pelas (t -1) operações sequenciadas;

S_t - Conjunto de operações sequenciáveis no estágio t, i.e. todas as operações que sucedem às que estão sequenciadas em Pt.

δ_k - A data mais cedo na qual a operação o_k em St poderia ser iniciada. Este tempo representa a conclusão de todos os precedentes de o_k e a disponibilidade de todos os recursos que o_k irá usar (recursos principais e recursos de suporte);

φ_k - A data mais cedo que a operação o_k em S_t pode ser finalizada, que é $\varphi_k = \delta_k + y_k$;

S^*_k - O conjunto de conflito formado por $o_k \in S_t$ que tem $\delta_k < \varphi^*$

o^*_k - A operação selecionada para ser sequenciada no estágio t .

Descrevem-se em seguida os passos do algoritmo:

Passo 1 - Seja t = 1 com P_1 nulo. S_1 será o conjunto formado por todas as operações que não têm predecessoras, ou seja pelas operações ligadas ao vértice inicial.

Passo 2 - Identificar $\varphi^* = \min \{ \varphi_k \}$ em Sk e a máquina M* na qual φ^* ocorre.

Passo 3 - Selecionar a operação o_k em S^*_t tal que necessita de M* e $\delta_k < \varphi^*$.

Passo 4 - Passar ao próximo estágio efetuando:

(1) aumentar a o_k a Pt e formar P_{t+1}.

(2) apagar ok desde St e formar St+1, adicionar a St a operação que se segue a ok na ordem de fabrico (com exceção se ok finalizar a ordem de fabrico),

(3) incrementar t de 1.

Passo 5 - Se existir alguma atividade ainda por sequenciar (t < nm), voltar ao *Passo 2,* senão parar.

O processo repete-se até não existirem mais operações para sequenciar.

Na **Figura 9** apresenta-se o momento da escolha da próxima operação a ser sequênciada usando o algoritmo GT.

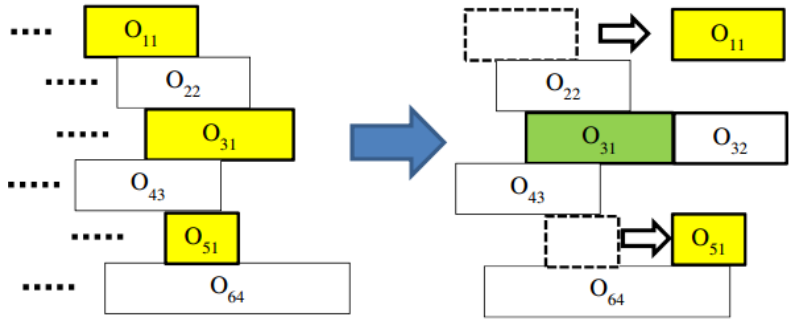


Figura 9 - Sequênciamento Giffler e Thomson

Em primeiro lugar escolhe-se a operação com menor tempo de término (O_{11}), ficando definida a máquina crítica (a amarelo). Em seguida escolhe-se aleatóriamente entre as operações sequenciáveis que são produzidas na máquina crítica, e possuem tempo de início menor que o tempo de término da operação O_{11}. Após a escolha da operação (O31), os tempos das restantes operações da máquina

crítica são actualizados (O11 e O51), e a próxima operação da tarefa entra na escolha da próxima iteração.

2.6.6 - Algoritmo Modificado de Giffler e Thompson

A principal especificidade do algoritmo modificado de Giffler e Thompson apresentado por French (1982) é o de produzir sequenciamentos não-atrasados. Se uma máquina estiver disponível e existir uma operação para processamento nessa máquina, então é iniciada a sua execução. Este objetivo é conseguido modificando os passos 2 e 3, isto é, escolhe-se o menor tempo de inicio possível, δ^*, do conjunto das operações sequenciáveis e escolhe-se a operação cujo tempo de início coincide com δ^*.

Passo 1 - Seja t = 1 com P_1 nulo. S_1 será o conjunto formado por todas as operações que não têm predecessoras, ou seja pelas operações ligadas ao vértice inicial.

Passo 2 - Identificar $\delta^* = \min \{ \delta_k \}$ em S_k e a máquina M* na qual δ_{k^*} ocorre.

Passo 3 - Selecionar a operação o_k em S^*_t tal que necessita de M* e $\delta_k < \delta^*$.

Passo 4 - Passar ao próximo estágio efetuando:

(1) aumentar a o_k a Pt e formar P_{t+1}.

(2) apagar ok desde St e formar St+1, adicionar a St a operação que se segue a ok na ordem de fabrico (com exceção se ok finalizar a ordem de fabrico).

(3) incrementar t de 1.

Passo 5 - Se existir alguma atividade ainda por sequenciar (t < nm), voltar ao *Passo 2,* senão parar.

2.7 - Meta-heurísticas

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

2.7.1 - Tabu Search

Este método inicialmente proposto por Glover (1977, 1986, 1989, 1990) é uma abordagem interativa que levou a várias formulações de sucesso para o problema job shop. É considerada uma técnica simples que de uma forma inteligente, com base na informação disponível, orienta o processo de procura afastando-se de soluções repetidas ou que se assemelham a soluções anteriormente alcançadas. Procedimentos mais elaborados podem ser aplicados para intensificar a procura em áreas que historicamente são de boa qualidade, ou então diversificam a procura em regiões não exploradas do espaço de soluções. A técnica de Nowicki e Smutnicki (1996) é atualmente uma das mais poderosas abordagens de tabu search permitindo que boas soluções sejam alcançadas rapidamente. Outros trabalhos importantes nesta área são de salientar, tais como de Dell'Amico (1993), Taillard (1994), Barnes e Chambers (1995), Lourenço e Zwijnenburg (1996) e Nowicki e Smutnicki (1996).

2.7.2 - Simulated Annealing (SA)

O Simulated Annealing é um método probabilístico proposto por Kirkpatrick, Vecchi e Gelett (1983) e Cerny (1985) de pesquisa do mínimo global de um espaço de soluções, que pode conter vários mínimos locais.

Este método é uma adaptação do algoritmo Metropolis-Hastings (Metropolis et al., 1953).

Na maioria dos algoritmos de optimização, existe uma procura de melhores soluções através da pesquisa de soluções na vizinhança de uma solução aleatória inicial. Se a solução vizinha é melhor que a atual, o algoritmo evolui nesse sentido. Todavia este processo pode levar a uma convergência prematura num mínimo local.

A propriedade fundamental do Simulated Annealing é a de ser possível aceitar uma solução pior que a atual para haver uma maior pesquisa do espaço de soluções. É esta propriedade que faz com que o algoritmo evite mínimos locais.

A probabilidade de aceitação de piores soluções, decresce ao longo das iterações do algoritmo, segundo uma determinada *temperatura de arrefecimento*.

Chama-se *temperatura de arrefecimento* à função que avalia a probabilidade de aceitação de piores soluções em cada iteração e tem esta nomenclatura devido a este algoritmo ter origem de um problema de arrefecimento de metais. Em cada iteração do algoritmo a temperatura diminui constantemente, até atingir o valor 0 ou um determinado número de iterações. A partir desse momento, o algoritmo apenas aceita soluções melhores que a atual.

Alterações a este método foram propostas por vários autores, nomeadamente Geman e Hwang, 1986; Gidas, 1985; Holley, Kusuoka e Stroock, 1989. Todavia a abordagem realizada neste trabalho tem por base o método clássico.

Neste projeto pretende-se aplicar o Simulated Annealing em conjunto com outro método de optimização, o Algoritmo Genético, e comprovar se se verificam melhorias de soluções.

2.7.3 - Algoritmo Genético

Os algoritmos genéticos tiveram origem nas décadas de 60 e 70 com o trabalho desenvolvido por uma equipa de investigadores da universidade do Michigan liderada por John Holland, onde resultou a publicação do livro *Adaptation in Natural and Artificial Systems* (1975). A sua investigação focou-se na abstração e explicação dos processos adaptativos naturais dos sistemas e no desenho de sistemas artificiais de software que retenham mecanismos importantes dos sistemas naturais. Esta abordagem levou a descobertas importantes na área científica dos sistemas naturais e artificiais. De Jong (1985) combina a teoria dos Schemata de Holland realizando importantes testes computacionais. Outros trabalhos sobre o tema foram realizados por Beasley et al. (1993) e Reeves (1991, 1993) demonstrando as vantagens e dificuldades de aplicação.

O algoritmo genético é um método bem aceite pela comunidade científica devido à sua simplicidade computacional e por ser uma poderosa ferramenta de pesquisa de melhoramento de soluções. Permite a inclusão de restrições e funções objetivo complexas mais próximas dos sistemas reais. Além destes motivos, esta técnica é de fácil modelação e integração com outras heurísticas, especialmente de pesquisa local (Vose, 1993).

Genericamente considera-se que um Algoritmo Genético é um modelo com base numa população de indivíduos que utilizam operadores de seleção e recombinação para explorar novos pontos no espaço de soluções, John Holland (1975).

Este método caracteriza-se por em cada iteração (geração) utilizar um conjunto de soluções (população) e cada solução representa um indivíduo dessa população (Gauthier, 1983). Em paralelo com os sistemas reais, existe uma seleção e reprodução dos indivíduos que de uma forma evolutiva, partindo de um conjunto de soluções mais simples se formem soluções de melhor qualidade (indivíduos mais aptos).

A implementação de um AG genérico inicia-se com a geração (tipicamente) aleatória dos cromossomas (população). Em seguida estes são avaliados e selecionados segundo uma dada probabilidade para se reproduzirem. Os cromossomas que representem uma melhor solução segundo o problema a resolver, têm uma maior probabilidade para se reproduzirem, do que os cromossomas que representem pior solução.

Algumas modificações ao AG clássico foram realizadas para colmatar algumas das suas limitações. O modelo matemático originalmente apresentado por Vose e Liepins (1991), pretendem dar resposta ao problema de convergência e distribuição da população ao longo do tempo.

No capítulo 3 deste trabalho, são descritos os operadores genéticos e parâmetros implementados, típicos do AG clássico.

2.7.4 - Hibridização de Algoritmos Genéticos (AG) e Simulated Annealing (SA)

Existem várias abordagens para a resolução de problemas de otimização, entre as quais com recurso a algoritmos genéticos e simulated annealing. Estas metodologias pesquisam iterativamente uma solução no espaço. Todavia existem algumas diferenças entre elas (Elhaddad, 2012).

Os AG trabalham sobre um conjunto de soluções (população) e trocam informação entre elas recorrendo a operadores genéticos (cruzamento/mutação). Estes utilizam o mesmo método de seleção de soluções durante todo o algoritmo.
Pelo contrário, o SA trabalha sobre uma solução de cada vez, e há uma atualização da temperatura de aceitação de soluções.

Enquanto que o SA pesquisa uma nova solução através da modificação de uma única solução local, o AG procura novas soluções através da combinação de várias soluções diferentes (cruzamento).

Estas diferenças são fundamentais no processo de escolha da melhor metodologia a adoptar. Existem vantagens e desvantagens em ambas, e a escolha deve ter em conta o problema a resolver, bem como a sua representação.

Algumas desvantagens das duas abordagens são conhecidas. No AG a possibilidade de convergência num mínimo local ou no elevado tempo necessário para encontrar a solução óptima (Houpt, 1995). Se a função de cruzamento não for a mais adequada para o problema, a solução resultante será maioritariamente aleatória (Liepins e Vose, 1991). No SA, não existe uma visão global do espaço de pesquisa devido a trabalhar com apenas uma solução candidata em cada iteração.

Para encontrar a solução óptima, o SA normalmente necessita de uma temperatura inicial elevada, com um menor valor de 'arrefecimento' o que leva a um tempo elevado de processamento do algoritmo e de convergência de soluções.

A elevada eficácia do AG em pesquisar grandes volumes no espaço de solução e a eficiência do SA em procurar melhores soluções vizinhas de uma solução local, é de prever que através da sua combinação nos permita melhorar a qualidade das soluções obtidas.

2.8 - Outras Abordagens

2.8.1 – Particle Swarm Optimization (PSO)

A optimização por exame de partículas (PSO) é um método de otimização estocástico desenvolvido por Eberhart e Kennedy em 1995, com base no comportamento social de bandos de pássaros ou cardumes de peixes.

Este método apresenta semelhanças com métodos evolucionários tais como Algoritmos Genéticos.

O algoritmo tem por base uma população de soluções aleatórias. É realizada uma pesquisa de soluções na vizinhança de cada solução actual, com uma determinada aceleração e direção.

Todavia, ao contrário dos Algoritmos Genéticos, o método PSO não tem operadores genéticos tais como o cruzamento ou mutação. No PSO as soluções potenciais chamadas partículas, "voam" no espaço de soluções seguindo as atuais partículas ideais (Eberhart e Kennedy, 1995).

Cada partícula mantém o registo (memória) das suas coordenadas no espaço de procura, que está associado à sua melhor solução encontrada (fitness). Ao longo das iterações, cada partícula procura soluções vizinhas melhores (pBest), identificando-se a região do espaço de soluções que se obteve a melhor resultado(lBest). Quando uma partícula leva toda a população para o seu local de pesquisa, a melhor solução encontrada é um óptimo global (gBest).

Este método consiste na procura de soluções em cada instante, alterando a velocidade de aceleração de cada partícula em direção à sua pBest para os locais lBest. Normalmente a aceleração toma valores aleatórios (Kennedy, 1997).

Nas últimas décadas, a PSO foi aplicada com sucesso em várias áreas de investigação, obtendo melhores resultados e mais rápidos que outros métodos de otimização (Jenigiri, 1996).

Uma das vantagens da PSO é no reduzido número de parâmetros que é necessário ajustar. Com poucas modificações do algoritmo, este ajusta-se a um grande número de problemas.

Comparativamente com os Algoritmos Genéticos (AG), o mecanismo de partilha de informação do PSO é significativamente diferente. No AG os cromossomas partilham a informação entre si (cruzamento). Deste modo toda a

população evolui segundo uma área ideal. No PSO, as partículas movem-se apenas num único sentido, isto é, segundo a informação do gBest ou lBest (Jenigiri, 1996).

Existem algumas limitações associadas ao PSO. Este método facilmente sofre com um optimismo parcial, que faz com que a sua velocidade e direção sejam permeáveis a erros. Outra limitação é na impossibilidade de aplicação em problemas onde não sejam possíveis modelar com recurso a coordenadas. (*Yonggang, Fengjie e Jigui*, 2006).

Em suma, o PSO é um novo método de otimização heurística que tem por base a inteligência das partículas. Comparativamente com outros algoritmos, este método é de simples implementação e ajustamento dos parâmetros. Contudo, a investigação utilizando este método ainda está pouco desenvolvida, pelo que ainda existem poucos resultados que comprovem os seus benefícios relativamente a outros métodos (Jenigiri, 1996).

3. Algoritmo Genético

3.1 - Representação de dados

É fundamental definir uma estrutura de dados, que irá representar cada solução candidata sobre a qual o Algoritmo Genético (*AG*) irá trabalhar adequadamente. A solução é representada pelo cromossoma.
Um indivíduo representa um ponto do espaço de pesquisa entre todas as soluções possíveis para o problema. O conjunto de cromossomas forma a *população*.

Dentro dos tipos de representação de cromossoma existem três que mais se destacam, a representação binária, inteira ou real (Bruns, 1993). A essa representação dá-se o nome de alfabeto do AG (Vose, 1999).

Cromossoma binário:	*(1001111010)*
Cromossoma real:	*(2.344, 1.01, 5.2112, 3.1276)*
Cromossoma inteiro:	*(6, 1, 10, 13, 9, 18, 3)*

A representação mais antiga é a binária, onde cada individuo é formado por uma cadeia de bits, que podem assumir os valores de 0 e 1. Esta representação é mais fácil de usar e manipular, pois facilita o uso dos operadores genéticos mais convencionais (Ex. *crossover/mutação)*. Por outro lado, para a optimização de valores contínuos, a representação binária requer um maior custo de execução devido a necessitar de muitos bits para obter uma boa representação numérica (Goldberg, 1989).

A representação com números inteiros é particularmente útil em problemas de permutação/ordenação, representando a sequência de objetos, sejam nodos num grafo ou operações de escalonamento. Estes problemas de permutação são típicos em otimização combinatória.

A representação com números reais é mais apropriada em problemas mais complexos, contudo necessita de operadores genéticos específicos para o problema (Goldberg, 1984).

A abordagem adotada neste projeto é a representação por Random Keys. É uma estratégia intermédia que combina a flexibilidade da representação binária e a aptidão sobre os problemas de sequenciamento da representação inteira.

É uma representação que se adequa a problemas combinatórios sem precisar de crossover específicos, o que a torna uma mais valia na aplicação ao problema JSSP (Pinedo, 2002).

Nesta representação os genes dos cromossomas são gerados aleatoriamente no intervalo entre [0 ,1] ou convertidos em inteiros.

3.2 - Fitness

Num AG, cada um dos cromossomas é avaliado por uma função de aptidão. Esta função determina a qualidade ou aptidão da solução representada por cada individuo.

Em alguns problemas usa-se a função objetivo do problema como sendo a função de avaliação dos cromossomas.

No caso do JSSP é usado a minimização do *makespan* com funções de avaliação. O valor da aptidão da solução ao meio em que se insere, o fitness, é fundamental para elaborar um ranking da população e consequentemente selecionar quais indivíduos podem participar no processo de reprodução. Os indivíduos são selecionados segundo uma probabilidade dada pelos seus fitness´s (Pinedo, 2002).

3.3 - Operações do AG

As operações elementares do AG são representadas na seguinte figura:

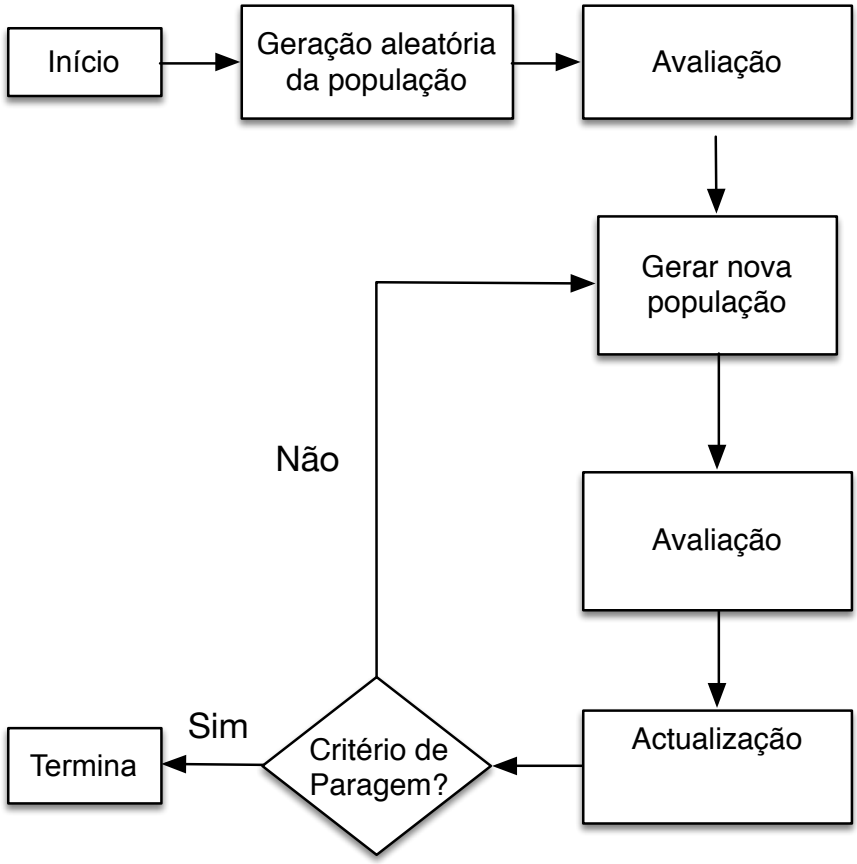


Figura 10 - Passos do AG
Fonte: Adaptado de Gonçalves, Mendes e Resende, 2005

Como podemos verificar na **Figura 10**, a etapa inicial do AG é a geração da população inicial P*(N)* com N indivíduos gerados aleatoriamente. Cada cromossoma representa uma possível solução para o problema, que será sujeita a uma avaliação.

Nessa fase de avaliação, a cada cromossoma é-lhe determinado um fitness, que representa a qualidade da solução que representa. No caso de estudo, este fitness é o resultado do algoritmo construtivo para a minimização do tempo de execução (*makespan*) de todas as tarefas de todas as ordens de fabrico.

Após a avaliação da solução de cada cromossoma é obtida uma nova população selecionando alguns indivíduos da população inicial e dos operadores genéticos (crossover/mutação).

A nova população é avaliada, e é feita uma atualização da população anterior, selecionando alguns indivíduos da população nova para substituir alguns indivíduos da população anterior.

Posteriormente volta-se a gerar uma nova população após a seleção de alguns indivíduos da população antiga e a aplicação dos operadores genéticos e assim sucessivamente.

A última etapa é a verificação do critério de paragem. O critério de paragem pode ser em função do número de iterações, tempo limite ou ter atingido a solução ótima. Caso tenha sido satisfeito o critério de paragem, o algoritmo termina com o resultado do melhor fitness dos indivíduos, identificando qual é o melhor cromossoma e a correspondente solução para o problema. Se não tiver sido satisfeito o critério de paragem, é gerada uma nova população e começa uma nova iteração do AG, até o critério de paragem ser satisfeito (Goldberg, 1989).

Nas secções seguintes é descrito em pormenor cada etapa do AG, bem como os métodos e técnicas implementadas para o melhor ajuste do AG ao projeto proposto.

3.3.1 - Inicialização da População

Na inicialização da população é definido previamente o número de indivíduos da população. Quanto maior for a população, maior será a diversidade genética e consequentemente maior o espaço de soluções e maior tempo de computação. Os cromossomas são gerados aleatoriamente com genes de valor aleatório (Goldberg, 1987).

A representação do cromossoma que se modelou neste projeto tem por base o conceito dos Random Keys. Esta estrutura de cromossomas apresenta uma grande aptidão para os problemas de sequenciamento como o JSSFP, e tem uma grande flexibilidade em problemas combinatórios, sem precisar de operadores genéticos específicos. É uma abordagem intermédia entre os cromossomas binários e os cromossomas reais (Gonçalves, Mendes e Resende, 2005).

Por questões de simplicidade e de melhor desempenho das linguagens de programação utilizadas (C / C++) adotou-se a seguinte alteração ao modelo convencional dos Random Keys. Em vez de se usar floats [0, 1[escolhe-se um valor inteiro conveniente multiplicando-se os genes por esse valor e usa-se a parte inteira.

Na **Figura 11** apresenta-se um cromossoma que resultou de um Random Key convencional [0, 1[multiplicado por 500.

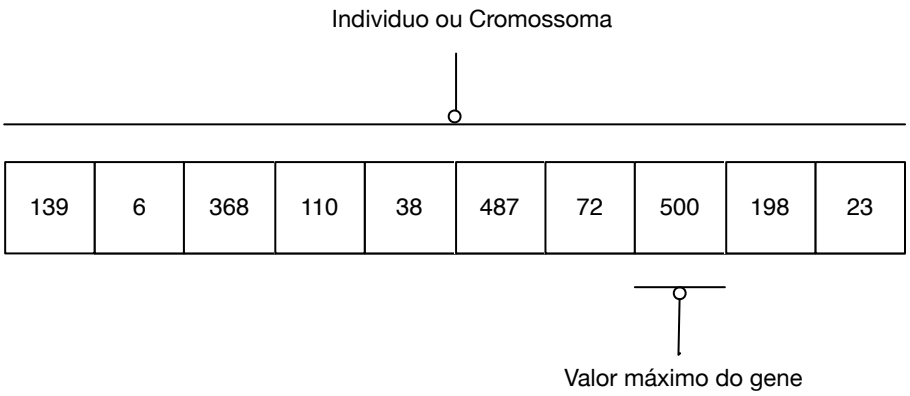


Figura 11 - Cromossoma aleatório, com valor máximo = 500

3.4 - Parâmetros genéticos

É importante fazer uma análise da influência que alguns parâmetros do AG têm na evolução da população. O ajuste destes parâmetros permite que seja melhorado o comportamento do AG para a resolução do problema proposto, de acordo com as restrições e recursos inerentes ao JSSP (Applegate e Cook, 1991).

3.4.1 - Tamanho da população

O tamanho da população determina o número de indivíduos da população. Quantos mais cromossomas, maior será a diversidade genética, aumentando o espaço de soluções. Contudo é determinante para a eficiência e desempenho do AG. Quanto maior for a população maior será o tempo de computação do algoritmo (De Jong et al., 1990).

Para populações pequenas o AG precisará de menor tempo de computação, contudo terá pior desempenho devido ao espaço de soluções ser menor e não haver uma diversidade genética que permita encontrar melhores soluções (Mitchell, 1996).

As populações grandes geralmente fazem uma melhor cobertura do espaço de soluções, evitando assim a convergência para mínimos locais. Todavia são necessários maiores recursos e tempo computacionais.

3.4.2 – Proporção de população cruzada

A proporção de população cruzada, representa a quantidade de indivíduos que serão gerados por cruzamento na nova população.

Para taxas de cruzamento de população grandes, mais rapidamente novos cromossomas serão introduzidos em cada iteração do AG, e se esta taxa for demasiado grande corre-se o risco da maior parte da população ser substituída e consequentemente perder-se soluções de maior aptidão (Goldberg, 1989).

Para taxas de cruzamento pequenas, menor será a evolução genética em cada iteração do AG, resultando num algoritmo lento.

3.4.3 – Taxa de cruzamento

A taxa de cruzamento é o operador responsável pela recombinação de características dos progenitores durante o cruzamento. Permite que as próximas gerações herdem as suas características.

Para taxas de cruzamento demasiado elevadas, a população descendente irá ser geneticamente semelhante aos seus progenitores, levando a uma convergência mais rápida.

Taxas de cruzamento demasiado baixas, implicam uma menor evolução que leva a que o algoritmo seja demasiado lento (Goldberg, 1989).

A taxa de cruzamento deve ser maior que a taxa de mutação.

3.4.4 - Tipo de cruzamento

O tipo de cruzamento determina o método da troca de informação entre os pares de cromossomas selecionados para se cruzarem.

Existem vários tipos de cruzamento, que devem ser aplicados de acordo com o problema que se pretende resolver. Esta troca de informação entre os cromossomas é fundamental para que a procura de soluções evolua globalmente (Schmitt, 2001).

É descrito mais à frente os tipos de cruzamentos que se utilizaram no projeto proposto.

3.4.5 - Taxa de mutação da população

A taxa de mutação da população determina a quantidade de cromossomas que serão sujeitos a sofrer algum tipo de mutação, em cada iteração do AG.

Para taxas de mutação da população altas corre-se o risco do AG ficar mais lento pois a evolução dos resultados obtidos nas iterações anteriores poderá perder-se ao mudar o valor dos genes mutados.

Com taxas de mutação mais pequenas, corre-se o risco de convergência prematura das soluções em mínimos locais, que origina a estagnação num dado valor (Goldberg, 1989).

3.4.6 - Taxa de mutação do cromossoma

A taxa de mutação do cromossoma determina a percentagem de genes que serão mutados. Previne que haja uma saturação de indivíduos semelhantes na população e consequentemente que haja convergência para mínimos locais.

Deve ser ajustado com especial cuidado para que não seja um impedimento na evolução das soluções mas também que permita uma maior diversidade genética.

3.4.7 - Critério de paragem

Determina em que condições se considera que o algoritmo genético encontrou uma solução aceitável para o problema ou tenha falhado na procura e não faça sentido continuar.

Existem vários critérios de paragem que dependem sobretudo do problema que se pretende resolver.

Tempo computacional, número de gerações, número de iterações sem que ocorra uma melhoria da melhor solução já obtida, obtenção de um valor pré estabelecido em relação à solução ótima ou um limite para a solução ótima são alguns exemplos de critérios de paragem.

3.5 - Operadores Genéticos

São os operadores genéticos que realizam a transformação da população através das várias gerações (Fox e McMahon, 1991).

Um bom operador genético é responsável pela propagação das boas características dos indivíduos contribuindo para a evolução da solução.

Os principais operadores genéticos do AG clássico são (Goldberg, 1989):

Seleção – realiza o processo de sobrevivência dos indivíduos mais aptos;
Cruzamento – realiza a troca de informação entre os indivíduos;
Mutação – introduz informação genética aleatória.

Uma estratégia bastante utilizada nas abordagens evolutivas é a do elitismo cujo conceito é a de prolongar a vida das melhores soluções (Vose, 1999).

Em seguida é descrito pormenorizadamente os operadores genéticos utilizados na aplicação desenvolvida neste trabalho.

3.5.1 - Seleção

O processo de seleção no AG é semelhante ao processo de sobrevivência na natureza, isto é, os indivíduos mais adaptados ao seu meio têm maior probabilidade de se reproduzirem do que os indivíduos considerados mais fracos.

No AG o processo de seleção possibilita aos cromossomas mais aptos uma maior probabilidade de se cruzarem e assim transmitirem a sua informação genética às próximas gerações.

3.5.1.1 - Roleta

Cada cromossoma tem uma probabilidade de ser selecionado para reprodução proporcional à sua aptidão.

A **Figura 12** representa um círculo divido em N regiões (tamanho da população) cuja área é proporcional à aptidão de cada individuo.

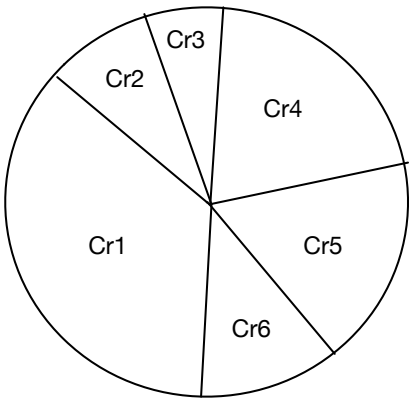


Figura 12 - Roleta

Após girar a roleta (através do lançamento de um número *random*), obtém-se o individuo selecionado. Os indivíduos mais aptos, têm uma região da roleta maior e assim, maior probabilidade de serem selecionados várias vezes.

Consequentemente a seleção dos indivíduos pode conter várias repetições de um só individuo. Do mesmo modo há outros que poderão não ser selecionados.

3.5.1.2 - Torneio

Existem outros métodos de seleção, como por exemplo seleção por torneio.

Neste método são selecionados aleatoriamente dois cromossomas e o cromossoma mais apto é selecionado.

Tal como podemos ver na **Figura 13**, o cromossoma selecionado é o Cr 3, pois é o individuo com maior aptidão, num problema de maximização.

Cr 3	30	10	57	90	15	3	65	Aptidão = 112
Cr 8	67	20	40	67	40	30	28	Aptidão = 109

Figura 13 - Torneio

Neste projeto combinou-se os dois métodos de seleção, onde uma percentagem de indivíduos são selecionados por roleta e a outra percentagem por torneio.

3.5.2 - Cruzamento

Após a seleção dos cromossomas, inicia-se a aplicação de um dos principais operadores genéticos, o crossover.

É nesta fase que haverá trocas de segmentos entre pares de cromossomas, que irão dar origem a novos indivíduos para formar a próxima geração.

O tipo de crossover nos algoritmos genéticos depende da representação das variáveis e do tipo de problema que se pretende resolver. Todavia existem dois métodos mais comuns de cruzamento:

3.5.2.1 - Cruzamento binário

No cruzamento binário utilizam-se dois progenitores para a formação de um descendente, usando um ou mais pontos de quebra dos pares de cromossomas progenitores. Para isso trocam-se os segmentos com informação dos dois indivíduos e originam-se dois descendentes distintos (Holland, 1992).

Salientam-se dois tipos de cruzamento binário:

3.5.2.2 - Single-point crossover

Um dos métodos mais simples, é o single-point crossover que estabelece aleatoriamente um ponto de corte em cada um dos progenitores gerando duas caudas e duas cabeças que são recombinadas para formar um descendente (**Figura 14**).

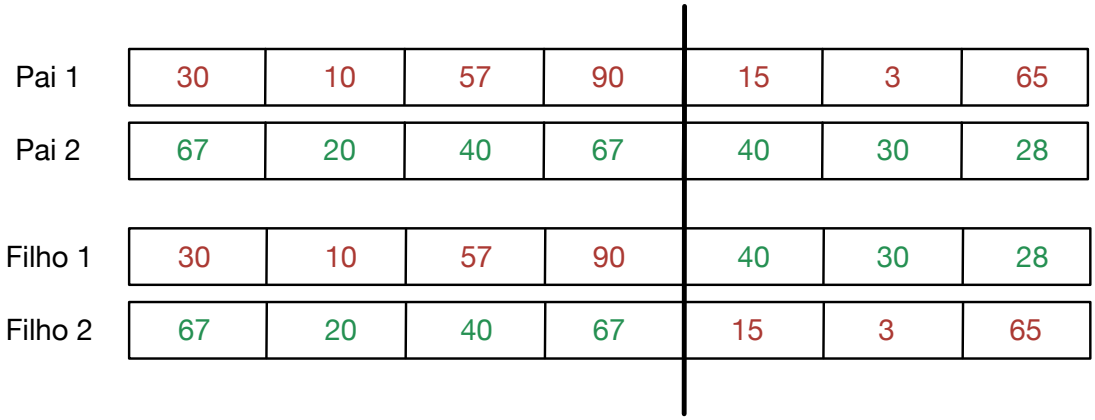


Figura 14 - Single-point crossover

3.5.2.3 - Uniform crossover

O método de cruzamento uniforme utiliza uma chave de números aleatórios com o mesmo comprimento do cromossoma, onde cada gene dos progenitores tem uma determinada probabilidade de pertencer ao descendente.

Como demonstra a **Figura 15**, para uma taxa de crossover de 60%, é gerada a chave A. Se o valor da chave em cada gene for inferior à taxa de crossover, o descendente terá o valor do *Progenitor 1*, se for inferior terá o valor do *Progenitor 2*.

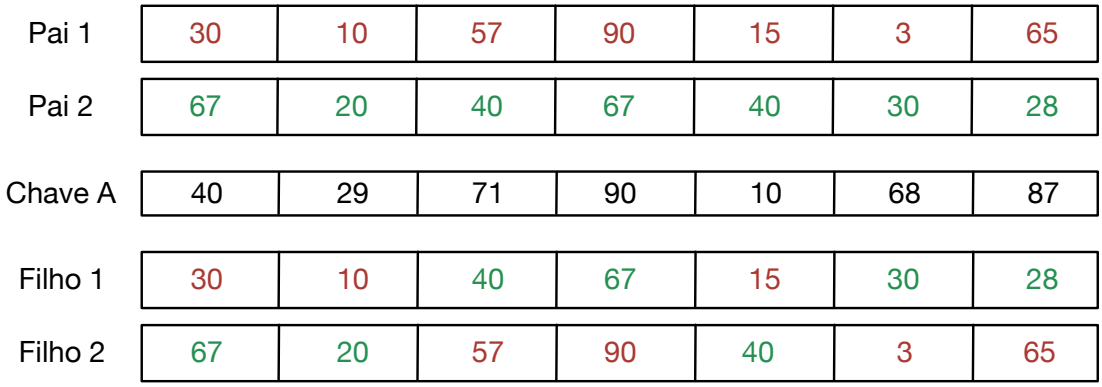


Figura 15 - Uniform crossover

3.5.2.4 - Cruzamento unário

No cruzamento unário existe um ajuste dos cromossomas dos progenitores que aumenta ou diminui o seu valor, segundo uma taxa de cruzamento.

Na **Figura 16**, vemos que se o valor de uma chave gerada aleatoriamente for superior à taxa de cruzamento de 60% então o gene do Filho1 será igual ao do Pai 1 e do Filho 2 igual ao do Pai 2.

Se o valor da chave for superior à taxa de cruzamento então há uma comparação entre os genes do Pai 1 e Pai 2. Se o gene do Pai 1 é superior ao Pai 2, dá origem a que o gene do Filho 1 tome o valor do Pai 1 menos uma unidade e o gene do Filho 2 toma o valor do Pai 2 mais uma unidade. Processa-se de forma inversa se o gene do Pai1 for inferior ao do Pai 2.

Deste modo há um reajuste de prioridades dos genes.

Pai 1	30	10	57	90	15	3	65
Pai 2	67	20	40	67	40	30	28
Chave A	40	29	71	90	10	68	87
Filho 1	30	10	56	89	15	4	64
Filho 2	67	20	41	68	40	29	29

Figura 16 - Cruzamento unário

3.5.3 – Mutação

Este operador genético troca aleatoriamente os valores de uma dada percentagem de genes do cromossoma. É responsável pela introdução e manutenção de alguma diversidade genética na população (Goldberg, 1989).

A fase da mutação é efetuada após o cruzamento, fornecendo a possibilidade da introdução de indivíduos diferentes dos seus progenitores. Contribui para que o espaço de soluções seja global evitando assim o problema da convergência em mínimos locais.

Entre os diferentes métodos de mutação, salientam-se a mutação binária e a mutação por permutação.

3.5.3.1 - Mutação Binária

Na mutação binária um gene de um descendente aumenta ou diminui aleatoriamente o seu valor (**Figura 17**).

Filho 1	30	10	56	89	15	4	64
Filho 1 Mutado	30	46	56	89	15	4	64

Figura 17 - Mutação Binária

3.5.3.2 - Mutação por Permutação

Neste tipo de mutação são selecionados aleatoriamente dois genes do indivíduo descendente e trocam-se entre si (**Figura 18**).

Filho 1	30	10	56	89	15	4	64
Filho 1 Mutado	30	64	56	89	15	4	10

Figura 18 - Mutação por Permutação

3.5.4 - Atualização da População

A atualização da população pode ser realizada de uma forma completa, onde toda a população descendente substitui a população que lhe deu origem. Os cromossomas modificados através dos operadores de crossover e mutação, são copiados para a população principal e posteriormente avaliados. Todavia não é garantido que estes sejam "melhores" que os cromossomas que lhes deram origem. Este tipo de atualização, pode comprometer o desempenho do AG na medida que se podem perder boas soluções.

Outra estratégia comum na atualização da população segue o paradigma de apenas os melhores indivíduos passam para a população principal. A população descendente é avaliada e a população final é o conjunto dos melhores indivíduos da população descente e progenitora. Trata-se de uma estratégia elitista que pode resultar numa convergência prematura para mínimos locais. Tem como pressuposto

que um individuo com menor ranking na geração atual, nunca poderá melhorar a solução nas próximas gerações (através dos operadores genéticos).

A estratégia implementada no projeto é intermédia às anteriores. Cada individuo descendente irá concorrer pelo lugar do seu progenitor. Formam-se clãs de indivíduos que apenas o melhor do clã passa para a população principal. Um progenitor pode ter vários descendentes numa dada geração, que após a avaliação o melhor irá competir com o progenitor na população principal (**Figura 19**).

Desta forma mantem-se uma diversidade genética na população e assim explorar todo o espaço de soluções.

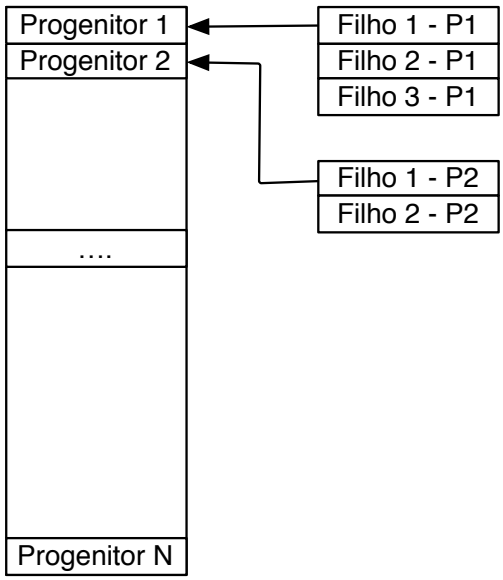


Figura 19 - Atualização da população

3.5.5 - Elitismo

Nos algoritmos genéticos clássicos, em cada geração da população é feita uma seleção da população e uma troca de informação entre os genes dos indivíduos.

Se os indivíduos não forem selecionados para reprodução, ou se a sua informação genética for destruída pelo crossover e mutação, pode levar a uma possível perda de um individuo com um elevado ranking genético.

Para melhorar a convergência dos AG e assim proteger as melhores soluções, desenvolveram-se métodos elitistas que visam transportar os melhores cromossomas

para as populações seguintes. Todos os métodos partilham da filosofia de prolongar a vida dos melhores cromossomas (Goldberg, 1989).

Estes métodos permitem um aumento no desempenho do AG, contudo contribuem para uma convergência prematura do algoritmo.

Um dos métodos clássicos de um algoritmo genético elitista, é de reservar um determinado número de posições em cada geração, para os melhores cromossomas da geração anterior. Consequentemente há uma seleção mais concentrada de progenitores pertencentes à elite. Nestes casos a elite deve tomar valores pequenos, caso contrário o algoritmo rapidamente converge para mínimos locais. O método adoptado no projeto contém métodos elitistas.

Tal como descreve a **Figura 20**, a elite é passada para uma população auxiliar. Esta população é constituída pela elite e pelos indivíduos provenientes da seleção.

A população descendente é formada por diferentes camadas de indivíduos com diferentes níveis de aplicação dos operadores genéticos. As posições iniciais contêm os indivíduos da elite mutados, seguidamente indivíduos apenas com crossover, crossover e mutação e por último apenas indivíduos mutados.

Finalmente, toda a população descendente é avaliada, e processa-se a atualização da população principal, onde os descendentes concorrem pelo lugar dos seus progenitores.

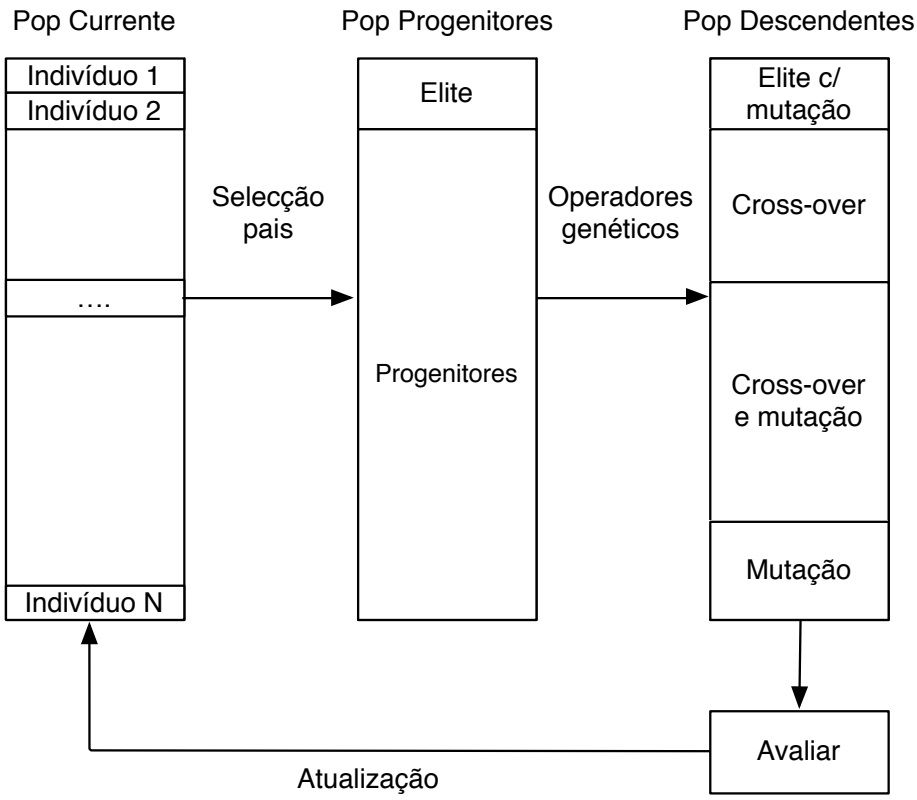


Figura 20 - AG com Elitismo

3.5.6 - População Backup

Foi implementada no projeto, uma estratégia inovadora que visa aumentar a diversidade genética da população e evitar uma rápida convergência prematura do algoritmo genético para mínimos locais.

Esta estratégia tem por base a formação de uma população backup, com o tamanho da população original. Esta população inicialmente é formada por indivíduos aleatórios, todavia irá sendo atualizada com os indivíduos que são substituídos pelos seus filhos na população descendente.

Caso existam mais do que um determinado número de cromossomas repetidos em cada geração, estes são substituídos pelos elementos da população backup.

Pretende-se com esta estratégia aumentar a diversidade de soluções, e assim evitar várias cópias do mesmo cromossoma pela população.

3.6 - Simulated Annealing

Origens

O método de Simulated Annealing tem como base uma técnica Metalúrgia, que envolve o aquecimento e arrefencimento controlado de um material. Essa técnica tem como objectivo reduzir defeitos através do aumento do tamanho dos cristais de um material (Granville et al., 1994).

O aquecimento e arrefecimento do material, afecta a temperatura e a energia termodinâmica livre. Embora a mesma quantidade de diminuição de temperatua provoque a mesma quantidade na diminuição da energia termodinâmica livre, se o arrefecimento for segundo uma dada taxa, irá provocar uma diminuição maior da energia livre.

Esta abordagem de diminuição de temperatura é implementado no algoritmo de Simulated Annealing, com uma lenta redução na probabilidade de aceitação de soluções piores na vizinhança.

Esta propriedade é característica do SA pois permite uma maior pesquisa dol espaço de soluções (Elhaddad, 2012).

Este método foi descrito por Kirkpatrick, Gelatt e Vecchi em 1983, e por Černý em 1985.

Descrição do método

Como podemos verificar no **Algoritmo 1**, o método inicia-se com uma solução aleatória x_p, e na selecção de uma solução vizinha x_n. Após o cálculo de custo das duas soluções é cálculada a sua diferença $\Delta f = f(x_n) - f(x_p)$. Se Δf for menor que zero, então o valor da função objectivo da solução vizinha é melhor, e substitui-se a solução corrente x_p por x_n.

Se Δf for maior ou igual a zero então a solução com pior resultado é aceite segundo uma probabilidade $pr = 1/(1 + e^{-\frac{\Delta E}{t}})$. O valor desta probabilidade diminui ao longo das iterações, onde *t* é a *temperatura de arrefecimento*. Se o valor de pr for maior que um número aleatório entre zero e um, então a solução com pior resultado é aceite (Elhaddad, 2012).

Este procedimento é repetido até que o critério de paragem seja satisfeito.

Algoritmo 1 – Passos do SA:

Criar a solução inicial x_p
Cálcular $f(x_p)$
Estabelecer t , $k = 1$
Estabelecer $0 < \beta < 1$.
Ciclo
  Seleccionar uma solução vizinha x_n
    Cálcular $f(x_n)$
    Cálcular $\Delta f = f(x_n) - f(x_p)$.
    Se ($\Delta f < 0$)
      Então $x_p = x_n$
      Senão
        Se $\frac{1}{1+e^{-\frac{-\Delta f}{t}}} > random\ (0,1)$
        Então $x_p = x_n$ Then
        Senão $t(k + 1) = \beta t\ (k)$
          $k = k + 1$

Até critério de paragem

Fonte: Adaptado de Elhaddad, 2012.

Simulated Annealing e Algoritmo Genético

Pretende-se com esta estratégia que o processo de substituição dos pais pelos filhos evolua segundo uma temperatura de arrefecimento que é atualizada à medida das gerações.

A implementação deste método estocástico estabelece uma probabilidade na substituição dos indivíduos, não sendo certo que nas gerações iniciais um filho com melhor fitness que o progenitor o substitua. Por outro lado um descendente com menor aptidão que o seu progenitor poderá substituir esse progenitor.

Tal como descreve a **Figura 21**, para gerações iniciais é estabelecida uma temperatura elevada, onde a probabilidade do filho substituir o pai é mais reduzida. À medida da evolução das gerações o método de substituição dos cromossomas passará a determinístico, onde a temperatura se irá reduzir até valores onde todos os melhores filhos substituem sempre os seus progenitores.

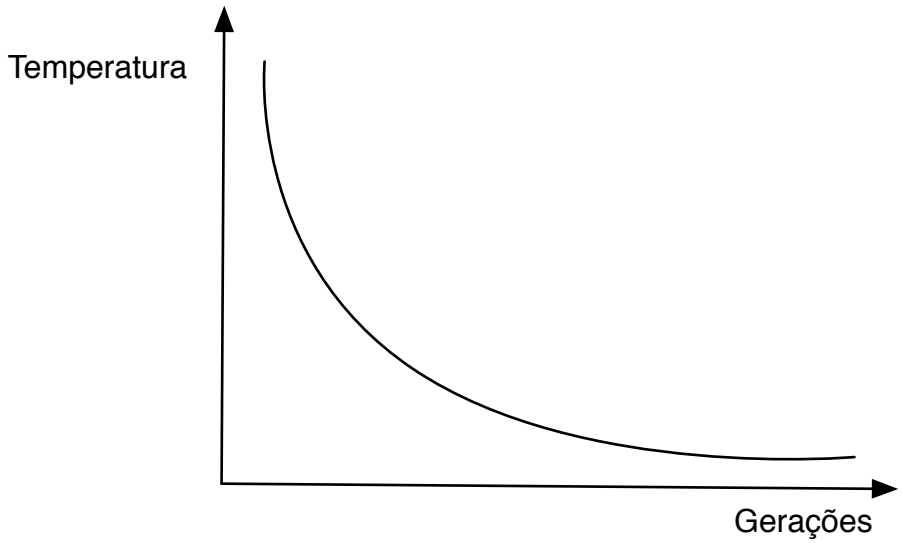


Figura 21 - Simulated Annealing

O objetivo da aplicação desta estratégia é dar mais aleatoriedade ao algoritmo genético evitando a rápida convergência para mínimos locais nas gerações iniciais e assim obter soluções de melhor qualidade.

4. Problema Job Shop

4.1 - Definição do problema

É reconhecido pela comunidade científica que o problema clássico de sequenciamento de operações job shop (JSSP) é extremamente difícil de solucionar (Boyd e Burlingame, 1996).

Existe um compromisso entre a eficiência em termos de tempo de computação e qualidade da solução (Bowman, 1959).

Na maioria dos métodos clássicos de otimização torna-se muito difícil encontrar uma solução ótima em tempo razoável, o que levou a adoptar um trabalho recorrendo a algoritmos genéticos, que fornece soluções aproximados em tempo de computação razoáveis.

O modelo job shop clássico define-se do seguinte modo (Gonçalves, Mendes e Resende, 2005):

Existem *n* jobs (ordens de fabrico) diferentes {J1, J2, ... , J*n*} para serem sequenciados em *m* máquinas diferentes {M1, M2, ... , M*m*}. A execução de um job i numa máquina j dá-se o nome de operação oij. A ordem de processamento das operações de um job nas máquinas é predefinida. Cada operação oij está associada a uma máquina onde é processada com um tempo de processamento pij.

Outras restrições dos jobs e máquinas são:

- um job não visita a mesma máquina duas vezes;
- não existem relações de precedência entre operações de jobs diferentes;
- uma operação que é iniciada, nunca é interrompida;
- cada máquina apenas pode processar um job de cada vez.

O objetivo do problema é de determinar a sequência de operações nas máquinas de forma a respeitar todas as restrições impostas, e optimizar o critério de performance. Geralmente este critério é o de minimizar o *makespan*, isto é, o tempo que leva a concluir todos os jobs, (Cheng, et al., 1996).

4.2 - Planos de sequenciamento

Destacam-se três tipos de planos válidos para o problema job shop:

Planos semi-ativos – nenhuma operação pode iniciar mais cedo sem alterar a ordem de processamento das máquinas.

Planos ativos - nenhuma operação pode iniciar mais cedo sem atrasar outra operação ou sem violar as restrições de precedência.

Planos não atrasados – nenhuma máquina é mantida em pausa se existir uma operação que pode ser iniciada nessa máquina.

Existem outros tipos de planos, contudo são considerados inválidos pois não respeitam as restrições a que o problema JSSP está sujeito.

Na **Figura 22** estão relacionados os tipos de planos de sequenciamento do JSSP. É possível verificar que todos os planos não atrasados são efetivamente planos activos, e que todos os planos activos são planos semi-activos.

O plano óptimo pertence ao conjunto de planos activos e dependendo da instância do problema poderá ser um plano não atrasado.

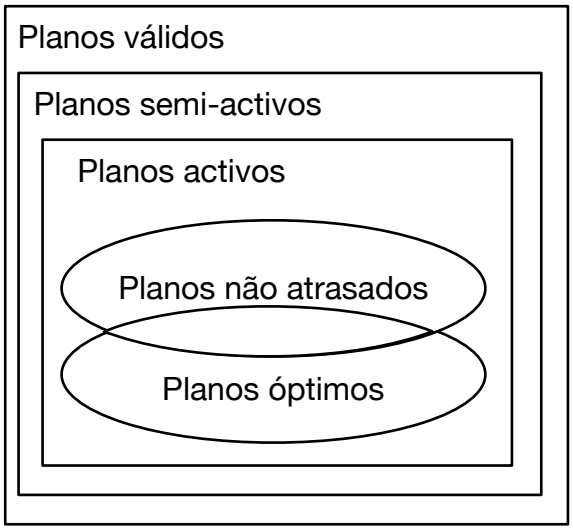


Figura 22 - Tipos de Planos

4.3 - Algoritmo Giffler and Thomson

Um método para gerar planos ativos é o algoritmo Giffler and Thomson. Este algoritmo tem uma notação específica que se descreve em seguida.

PS_n – plano parcial com n = {1.. N} operações planeadas. N representa o número total de operações

S_n – conjunto de operações que podem ser sequenciáveis em PS_n.

M_j – máquina onde é processada a operação *j*.

p_j – tempo de processamento da operação *j*.

σ_j – instante mais cedo em que se pode iniciar a operação *j*, com $j \in S_n$.

ϕ_j – instante mais cedo em que pode ser concluída a operação j, com $j \in S_n$.

ϕ^* – menor valor dos ϕ_j.

*M** – máquina de conflito, isto é, a máquina onde ocorre ϕ^*, ou seja, onde a próxima operação a ser sequenciada se irá processar.

*S**– conjunto de conflitos, isto é, operações que podem ser sequenciáveis na máquina *M**.

Em cada iteração é selecionada uma operação $j \in S^*$ a ser incluída em PS_n.

4.3.1 - Planos ativos

Algoritmo 1 – geração de planos activos de Giffler e Thompson

Passo 1 : seja n = 1, $PS_n = \{\}$. S_n o conjunto de todas as operações sem predecessores.

Passo 2 : encontrar $\phi^* = \min\{ \phi_{n \square Sn} \}$ e a máquina *M** onde o_j com ϕ^* é executada. Caso existam vários *M**, escolher aleatoriamente.

Passo 3 : escolher uma operação o_j do conjunto S^* onde:

- operação o_j necessita da máquina *M** ;
- $\sigma_j < \phi^*$;

Passo 4 : continuar com a próxima iteração atualizando:

- PS_n com a operação o_j resultando PS_{n+1}
- Remover o_j de S_n e formar S_{n+1} adicionando o sucessor de o_j (exceptuando se o_j for a última operação).
- n = n+ 1

- Atualizar σ_j e ϕ_j às operações de S_{n+1}

O algoritmo Giffler and Thomson gera planos válidos ativos. Em cada iteração, é formado um conjunto de conflitos S* no qual uma operação é escolhida para ser sequenciada. É repetido o processo até que todas as operações sejam sequenciadas, respeitando as ordens de precedência.

Nos passos 2 e 3 é formado o conjunto de conflitos, onde a operação selecionada irá atrasar o inicio das outras operações de S*. É esta a condição de um plano ativo, isto é, nenhuma operação pode iniciar mais cedo sem atrasar as outras.

4.3.2 - Planos não atrasados

Podem-se gerar planos atrasados fazendo uma modificação nos passos 2 e 3 do algoritmo de Giffler e Thomson. Deste modo assegura-se que nenhuma máquina fica parada quando existe uma operação pronta para ser executada naquela máquina.

Algoritmo 1.2 Modificado – geração de planos não atrasados de Giffler e Thompson

Passo 1 : seja n = 1, PS_n ={ }. S_n o conjunto de todas as operações sem predecessores.

Passo 2 : encontrar $\sigma^* = \min\{ \sigma_{n \square Sn} \}$ e a máquina M^* onde o_j com σ^* é executada. Caso existam vários M^*, escolher aleatoriamente.

Passo 3 : escolher uma operação o_j do conjunto S^* onde:

- operação o_j necessita da máquina M^* ;
- $\sigma_j = \sigma^*$;

Passo 4 : continuar com a próxima iteração atualizando:

- PS_n com a operação o_j resultando PS_{n+1}
- Remover o_j de S_n e formar S_{n+1} adicionando o sucessor de o_j (exceptuando se o_j for a última operação).
- n = n+ 1
- Atualizar σ_j e ϕ_j às operações de S_{n+1}

Com a aplicação do algoritmo de Giffler and Thomson para gerar planos não atrasados reduz-se significativamente o espaço de soluções, correndo o risco de se perder a solução ótima.

4.4 - Algoritmo Genético e Giffler e Thomson

No contexto do projeto implementado, os cromossomas do AG têm um papel importante na escolha das operações do conjunto de conflitos S*. Após selecionada a máquina crítica, se existirem várias operações que concorrem por M* o cromossoma tem o papel de desempate, isto é, a operação com maior prioridade no cromossoma será a escolhida para se juntar a PSn. Esta condição leva a que apenas se obtenham planos ativos ou não-atrasados aumentando a eficácia e eficiência do algoritmo.

4.4.1 - Geração de planos ativos

Exemplo 1

A **Tabela 3** representa um problema de 3 jobs a serem processados em 3 máquinas. O conjunto de jobs = { J_1 , J_2 , J_3 } tem uma sequência de processamento própria, a serem processados no conjunto m de máquinas { m_1 , m_2 , m_3 }. O processamento de um job j numa máquina m representa uma operação ojm. Cada operação utiliza exclusivamente a máquina em que está a ser processada durante uma duração p_m.

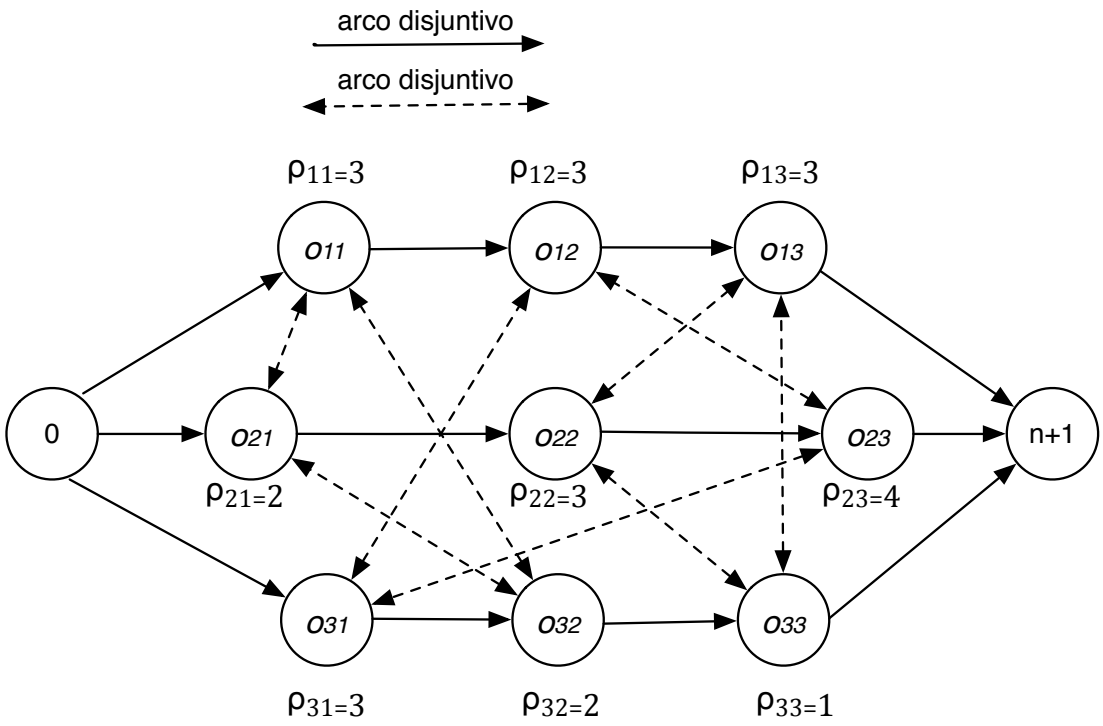
Tabela 3 - Problema 3 x 3

	(m, p)	*(m, p)*	*(m, p)*
job 1:	*(1 , 3)*	*(2 , 3)*	*(3 , 3)*
job 2:	*(1 , 2)*	*(3 , 3)*	*(2 , 4)*
job 3:	*(2 , 3)*	*(1 , 2)*	*(3 , 1)*

4.4.2 - Grafo disjuntivo

O problema JSSP demonstrado na **Tabela 3,** pode ser representado por um grafo disjuntivo $G = (V, C \cup D)$ da **Figura 23,** onde:

- V é o conjunto de vértices que representam as operações a serem processadas. O vértice 0 é o inicio do plano, e n+1 o final.
- C é o conjunto dos arcos conjuntivos que representam a ordem de precedência das operações.
- D é o conjunto dos arcos disjuntivos que representam pares de operações que correm na mesma máquina.



$o_{jm=}$ operação do job *j* na máquina *m*

$\rho_{jm=}$ tempo de processamento da operação

Figura 23 - Grafo disjuntivo do problema 3 x 3

O gráfico disjuntivo do problema 3 x 3 representa todas as possíveis ordens de processamento de todas as operações que são processadas na mesma máquina.

O problema consiste em transformar os arcos disjuntivos em arcos com apenas um sentido, isto é, obtém-se um plano válido através de um conjunto direcionado de arcos resultantes dos arcos disjuntivos. Este conjunto diz-se completo se todos os arcos disjuntivos forem selecionados, e consistente se o grafo resultante for acíclico.

Um plano semi-activo obtém-se através de um grafo completo e consistente onde as operações se iniciam o mais cedo possível (**Figura 24**).

O comprimento do maior caminho desde o vértice 0 até ao vértice n+1 resulta no *makespan*. Esse caminho designa-se de caminho crítico e é composto pelas operações críticas.

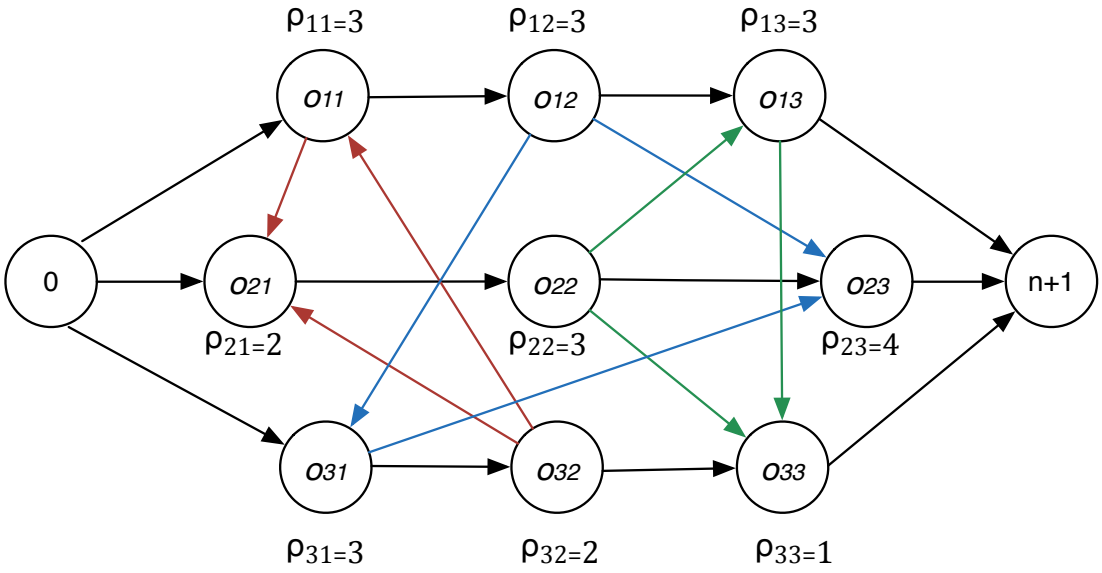


Figura 24 - Grafo disjuntivo do problema 3 x 3

O caminho mais longo entre os vértices 0 e n+1 (*caminho crítico*) é estabelecido pelas operações críticas o11, o12, o31, o23 com o valor de 13 unidades.

A **Figura 25** representa o gráfico de Gantt para o problema 3 x 3. O plano resultante é um plano semi-activo pois nenhuma operação se poderá iniciar mais cedo sem desrespeitar as restrições tecnológicas (precedências).

O makespan resultante tem o valor de 12 unidades.

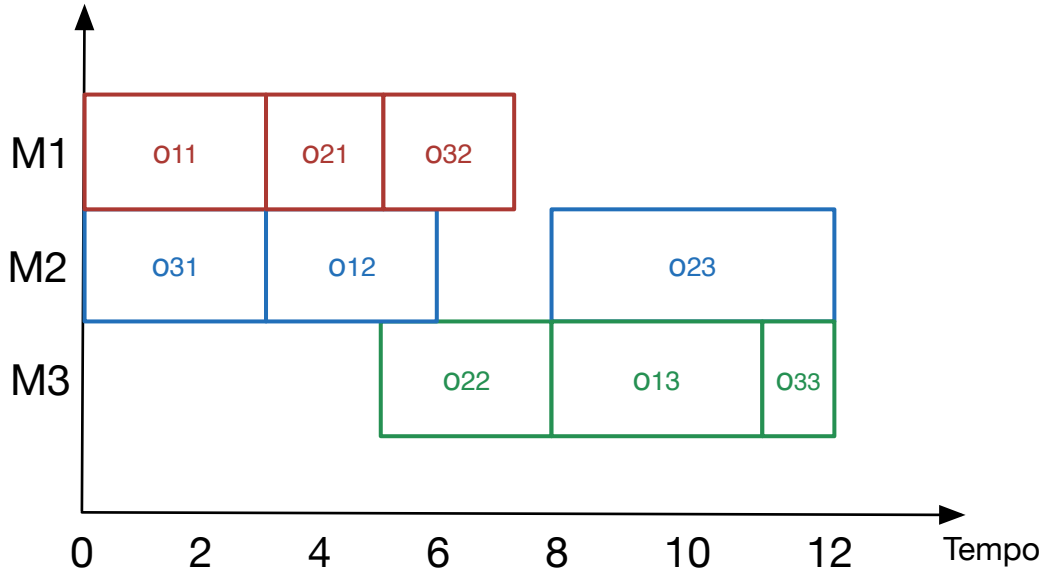


Figura 25 - Gráfico de Gantt – Plano semi-activo

4.4.3 - Aplicação do algoritmo Giffler e Thomson

Em seguida demonstra-se a geração de um plano ativo para o problema da **Tabela 5**, através da aplicação do algoritmo Giffler e Thomson.

À medida da formação do conjunto PSn formar-se-á o gráfico de Gantt da solução. O cromossoma exemplo é formado pelos seguintes genes (**Tabela 4**):

Tabela 4 - Cromossoma

Nº Operação	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
Prioridade Gene	3	1	9	6	2	5	8	4	7

Tabela 5 - Problema 3 x 3; i corresponde ao nº da operação

	(m, ρ, i)	*(m, ρ, i)*	*(m, ρ, i)*
job 1	*(1 , 3, 1)*	*(2 , 3, 2)*	*(3 , 3, 3)*
job 2	*(1 , 2, 4)*	*(3 , 3, 5)*	*(2 , 4, 6)*
job 3	*(2 , 3, 7)*	*(1 , 2, 8)*	*(3 , 1, 9)*

Apresenta-se, nas Tabelas e Figuras seguintes, a matriz de iteração do algoritmo:

Tabela 6 - Matriz iteração, S = 1

O_{ij}	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
S_1	|	-	-	|	-	-	|	-	-
M_i	1	-	-	1	-	-	2	-	-
σ_i	0	-	-	0	-	-	0	-	-
ϕ_i	3	-	-	2	-	-	3	-	-
ϕ^*	-	-	-	2	-	-	-	-	-
M*	-	-	-	1	-	-	-	-	-
S*	-	-	-	|	-	-	-	-	-

S_1 é formado por todas as operações sem predecessores = O_{11} , O_{21}, O_{31}

$\phi^* = \min(3, 2, 3) = 2$

M* = 1

S* = {O_{11} ; O_{21}} , segundo o critério de escolha da operação cujo gene tem maior prioridade, é selecionada a operação O_{21}

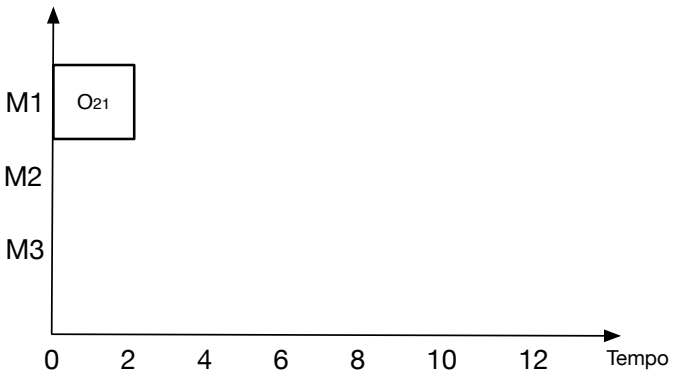


Figura 26 - Gantt, S = 1

Tabela 7 - Matriz iteração, S = 2

O_{ij}	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
S_1	|	-	-	\	|	-	|	-	-
M_i	1	-	-	\	3	-	2	-	-
σ_i	2	-	-	\	2	-	0	-	-
ϕ_i	5	-	-	\	5	-	3	-	-
ϕ^*	-	-	-	\	-	-	3	-	-
M^*	-	-	-	\	-	-	2	-	-
S^*	-	-	-	\	-	-	|	-	-

S_2 é formado pelo conjunto O_{11} , O_{22}, O_{31}

$\phi^* = \min(5, 6, 3)$

$M^* = 2$

$S^* = \{O_{31}\}$

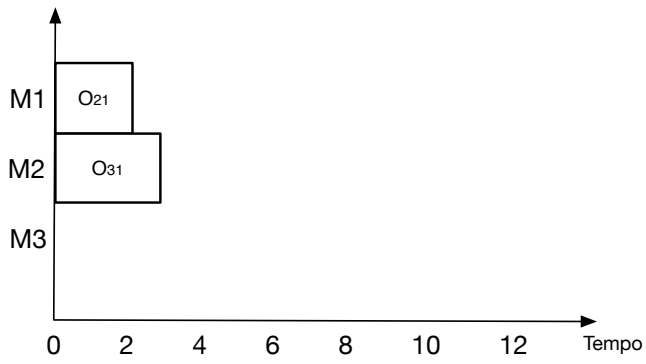


Figura 27 - Gantt, S = 2

Tabela 8 - Matriz iteração, S = 3

O_{ij}	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
S_1	|	-	-	\	|	-	\	|	-
M_i	1	-	-	\	3	-	\	1	-
σ_i	2	-	-	\	2	-	\	3	-
ϕ_i	5	-	-	\	5	-	\	5	-
ϕ^*	-	-	-	\	-	-	\	5	-
M*	-	-	-	\	-	-	\	1	-
S*	-	-	-	\	-	-	\	|	-

S_3 é formado pelo conjunto O_{11} , O_{22}, O_{32}

$\phi^* = \min(5, 5, 5)$; Consultando o cromossoma, a operação com maior prioridade é a O_{32} consequentemente é escolhida a operação O_{32} ;

$M^* = 1$

$S^* = \{O_{32}\}$

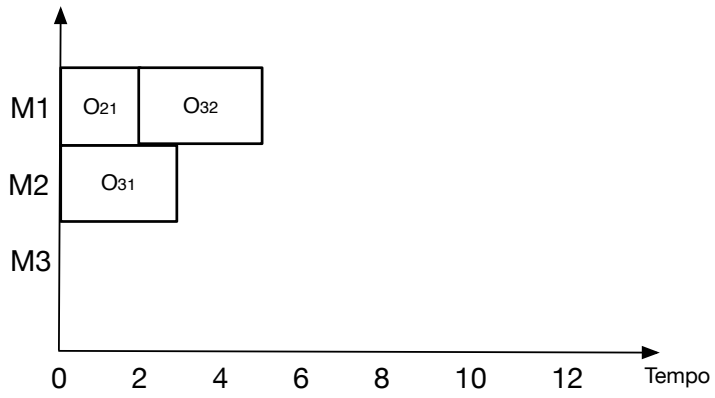


Figura 28 - Gantt, S = 3

Tabela 9 - Matriz iteração, S = 4

O_{ij}	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
S_1	|	-	-	\	|	-	\	\	|
M_i	1	-	-	\	3	-	\	\	3
σ_i	5	-	-	\	2	-	\	\	2
ϕ_i	8	-	-	\	5	-	\	\	3
ϕ^*	-	-	-	\	-	-	\	\	3
M*	-	-	-	\	-	-	\	\	3
S*	-	-	-	\	-	-	\	\	|

S_4 é formado pelo conjunto O_{11} , O_{22}, O_{33}

$\phi^* = \min(8, 5, 3) = 3$

M* = 3

S* = { O_{22}, O_{33}}, o gene com maior prioridade é da operação O_{33}

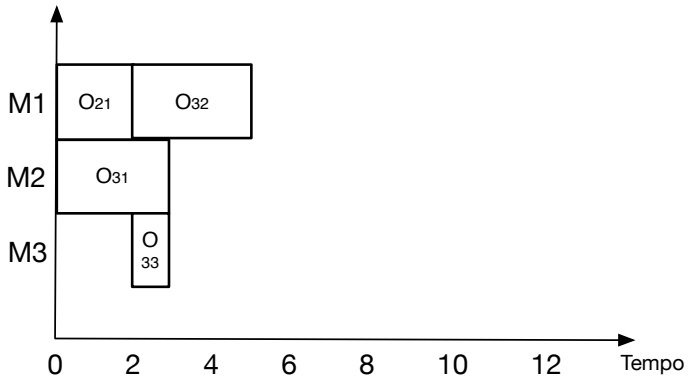


Figura 29 - Gantt, S = 4

Tabela 10 - Matriz iteração, S = 5

O_{ij}	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
S_1	|	-	-	\	|	-	\	\	\
M_i	1	-	-	\	3	-	\	\	\
σ_i	5	-	-	\	3	-	\	\	\
ϕ_i	8	-	-	\	6	-	\	\	\
ϕ^*	-	-	-	\	6	-	\	\	\
M*	-	-	-	\	3	-	\	\	\
S*	-	-	-	\	|	-	\	\	\

S_5 é formado pelo conjunto O_{11}, O_{22}

$\phi^* = \min(8, 6) = 6$

$M^* = 3$

$S^* = \{O_{22}\}$

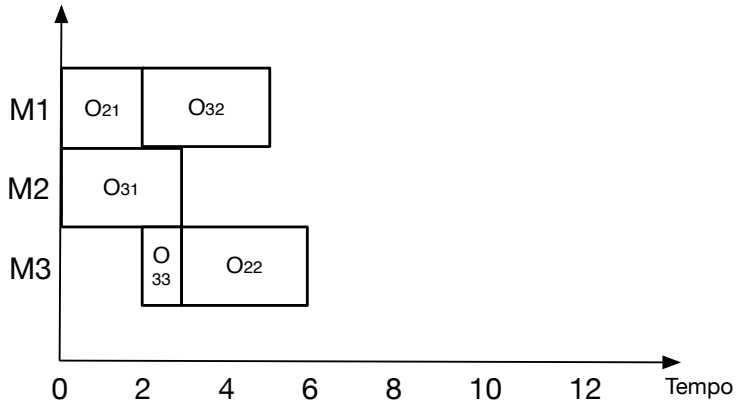


Figura 30 - Gantt, S = 5

Tabela 11 - Matriz iteração, S = 6

O_{ij}	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
S_1	|	-	-	\	\	|	\	\	\
M_i	1	-	-	\	\	2	\	\	\
σ_i	5	-	-	\	\	3	\	\	\
ϕ_i	8	-	-	\	\	7	\	\	\
ϕ^*	-	-	-	\	\	7	\	\	\
M^*	-	-	-	\	\	2	\	\	\
S^*	-	-	-	\	\	|	\	\	\

S_6 é formado pelo conjunto O_{11}, O_{23}

$\phi^* = \min(8, 7) = 7$

$M^* = 2$

$S^* = \{O_{23}\}$

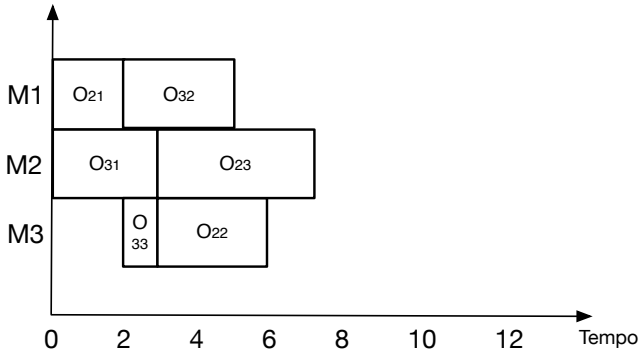


Figura 31 - Gantt, S = 6

Tabela 12 - Matriz iteração, S = 7

O_{ij}	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
S_1	|	-	-	\	\	\	\	\	\
M_i	1	-	-	\	\	\	\	\	\
σ_i	5	-	-	\	\	\	\	\	\
ϕ_i	8	-	-	\	\	\	\	\	\
ϕ^*	8	-	-	\	\	\	\	\	\
M*	1	-	-	\	\	\	\	\	\
S*	|	-	-	\	\	|	\	\	\

S_7 é formado pelo conjunto O_{11}

$\phi^* = \min(8)$

$M^* = 1$

$S^* = O_{11}$

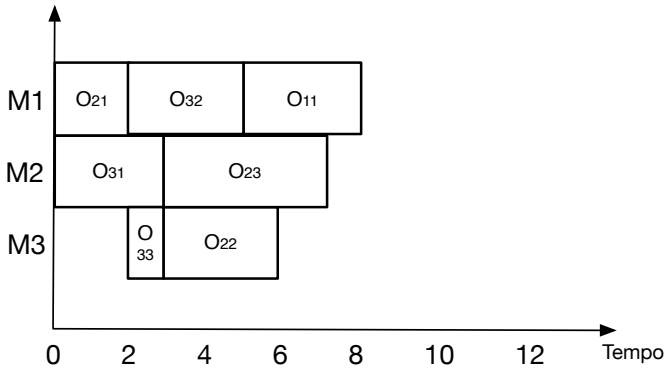


Figura 32 - Gantt, S = 7

Tabela 13 - Matriz iteração, S = 8

O_{ij}	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
S_1	\	|	\	\	\	\	\	\	\
M_i	\	2	\	\	\	\	\	\	\
σ_i	\	7	\	\	\	\	\	\	\
ϕ_i	\	10	\	\	\	\	\	\	\
ϕ^*	\	10	\	\	\	\	\	\	\
M*	\	2	\	\	\	\	\	\	\
S*	\	|	\	\	\	|	\	\	\

S_8 é formado pelo conjunto O_{12}

$\phi^* = \min(10)$

$M^* = 2$

$S^* = O_{12}$

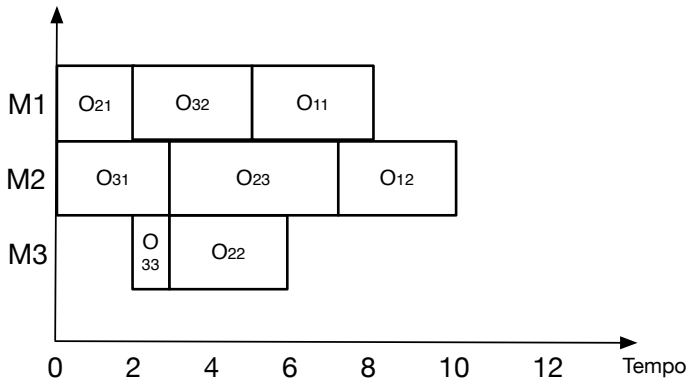


Figura 33 - Gantt, S = 8

Por fim, é escalonada a última operação O_{13} e é formado o gráfico de Gantt do plano ativo exemplo, com um makespan de 10.

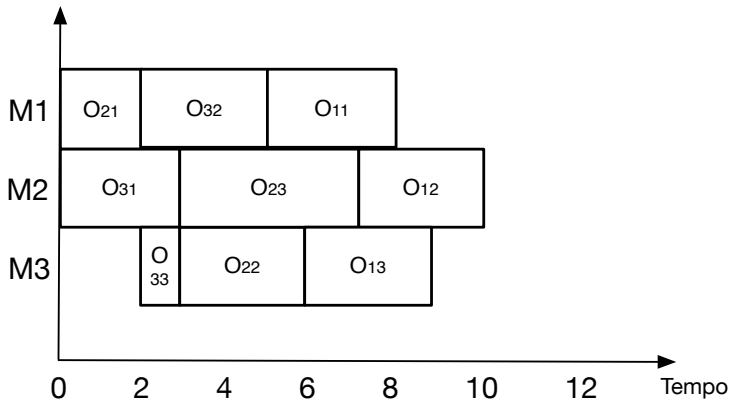


Figura 34 - Gantt, S = 9

5. Experiências Computacionais

Nesta secção são apresentados os resultados computacionais obtidos através da aplicação do AG desenvolvido para problema JSSP.

Os testes foram realizados sobre dois conjuntos de instâncias. O primeiro grupo é composto por duas instâncias (FT06, FT10) desenvolvidas por Fisher e Thompson (1963). O segundo grupo é composto por oito instâncias *(LA02, LA19, LA21, LA27, LA29, LA30, LA36, LA40)* desenvolvidas por Lawrence (1984).

Estas instâncias são as mais frequentemente utilizadas para testes de desempenho. Englobam diferentes tipos de problema com níveis de dificuldade diferentes e tamanhos variáveis segundo a seguinte tabela:

Tabela 14 - Ordens de Fabrico vs. Máquinas das instâncias de teste

Instância	*Ordens de Fabrico*	*Máquinas*
FT06	6	6
FT10	10	10
LA02	10	5
LA19	10	10
LA21	15	10
LA27	20	10
LA29	20	10
LA30	20	10
La36	15	15
LA40	15	15

Para a realização dos testes foram elaboradas diferentes configurações do AG.

Cada experiência tem uma configuração própria que pretende testar a influência dos parâmetros nos resultados obtidos.

Para cada instância foram realizadas 10 corridas. Analisa-se os resultados da Melhor Corrida, Corrida Média e Pior Corrida.

Para comparação de resultados entre as experiências foram selecionadas duas instâncias de referência *(FT10, LA36)*, que são de dimensão média e dificuldade elevada.

5.1 - Recursos tecnológicos

O algoritmo genético e algoritmo construtivo foram implementados em *C++* e os testes realizados num *MacBook Pro 13', 2.4GHz dual-core Intel Core i5* e sistema operativo *Mac Os Snow Leopard*.

Em cada experiência efectuada é explicado o parâmetro do AG que se vai testar bem como os resultados esperados.

Após a descrição dos resultados obtidos é feita uma análise crítica das soluções.

As instâncias utilizadas para comparação de experiências são a FT10 e LA36, ambas são de complexidade elevada.

5.2 - Experiências

5.2.1 – P1

Comparação dos resultados obtidos com diferentes tamanhos da população.

Nesta experiência são obtidos resultados para duas populações de tamanho diferente.

A configuração utilizada para o AG é a seguinte:

	População Pequena	*População Grande*
Tamanho da População	20	100
Taxa de Cruzamento	0,65	0,65
Elite	5%	5%
Indivíduos cruzados (%)	60	60
Indivíduos cruzados e mutados (%)	20	20
Individuos mutados (%)	20	20
Mérito	Min. Makespan	Min. Makespan
Critério de Paragem	500 iterações	500 iterações

Configuração 1 - Tamanho População Variável

Um dos parâmetros mais importantes de um AG é o tamanho da população. A qualidade das soluções obtidas, em muitos problemas, está relacionada com a correta configuração deste parâmetro.

Contudo quanto maior for a população maior será o custo computacional e o tempo de processamento.

Para instâncias mais simples, isto é, com menos ordens de fabrico e máquinas de processamento, não é necessário uma grande diversidade genética para se encontrar uma boa solução, devido ao facto da aplicação do algoritmo construtivo (Giffler e Thomson). Em casos de problemas cujo caminho crítico é de dificuldade reduzida, cromossomas geneticamente diferentes podem levar à mesma solução, devido ao uso do Giffler e Thomson para construção de soluções válidas.

Em instâncias de maior dificuldade o desempenho do AG é melhor, se houver uma diversidade genética grande e assim ser possível o AG aumentar a pesquisa no espaço de soluções. Contudo existe um maior custo de computação e eficiência.

Os resultados obtidos com a *Configuração 1* são demonstrados na **Tabela 15**.

Em cada instância são analisados os valores de:

- Solução Óptima *(S.O.);*
- Pior Corrida *(P.C.);*
- Corrida Média *(C.M.);*
- Melhor Corrida *(M.C.);*
- Distância Absoluta da Pior Corrida à Solução Óptima *(D1);*
- Distância Absoluta da Corrida Média à Solução Óptima *(D2);*
- Distância Absoluta da Melhor Corrida à Solução Óptima *(D3);*
- *% Variação Relativa da Pior Corrida entre Pop. Grande e Pop. Pequena (V1);*
- *% Variação Relativa da Corrida Média entre Pop. Grande e Pop. Pequena (V2);*
- *% Variação Relativa da Melhor Corrida entre Pop. Grande e Pop. Pequena (V3);*

Tabela 15 - Resultados com a Configuração 1

Instância	Tamanho Pop.	S.O.	P.C.	C.M.	M.C.	D1	D2	D3	V1	V2	V3
FT06	Pop. Grande	55	56	55	55	1	0	0	-	-	-
	Pop. Pequena	55	57	56	56	2	1	1	1,79	1,82	1,82
FT10	Pop. Grande	930	1060	1041	1018	130	111	88	-	-	-
	Pop. Pequena	930	1080	1068	1029	150	138	99	1,89	2,59	1,08
LA02	Pop. Grande	655	685	679	655	30	24	0	-	-	-
	Pop. Pequena	655	687	666	662	32	11	7	0,29	-1,91	1,07
LA19	Pop. Grande	842	930	926	913	88	84	71	-	-	-
	Pop. Pequena	842	947	942	918	105	100	76	1,83	1,73	0,55
LA21	Pop. Grande	1046	1258	1239	1209	212	193	163	-	-	-
	Pop. Pequena	1046	1287	1227	1191	241	181	145	2,31	-0,97	-1,49
LA27	Pop. Grande	1235	1507	1498	1466	272	263	231	-	-	-
	Pop. Pequena	1235	1533	1508	1472	298	273	237	1,73	0,67	0,41
LA29	Pop. Grande	1152	1416	1403	1381	264	251	229	-	-	-
	Pop. Pequena	1152	1434	1422	1418	282	270	266	1,27	1,35	2,68
LA30	Pop. Grande	1355	1554	1525	1517	199	170	162	-	-	-
	Pop. Pequena	1355	1887	1547	1521	532	192	166	21,43	1,44	0,26
LA36	Pop. Grande	1268	1622	1526	1515	354	258	247	-	-	-
	Pop. Pequena	1268	1557	1543	1523	289	275	255	-4,01	1,11	0,53
LA40	Pop. Grande	1222	1494	1455	1428	272	233	206	-	-	-
	Pop. Pequena	1222	1507	1469	1432	285	247	210	0,87	0,96	0,28

5.2.1.1 - Análise de resultados

Nas instâncias mais simples, FT06, LA02 e LA19 as soluções encontradas estão próximas da solução óptima. É esperado que com o aumento do número de iterações do AG, sejam encontrados os valores óptimos.

Na generalidade das instâncias, há um aumento de desempenho com o aumento do tamanho da população. Verifica-se apenas na instância LA21 uma ligeira diminuição de desempenho, que poderá ser característica da instância. Uma menor população permite uma maior intensificação em torno de mínimos locais. Esta situação é frequente em problemas difíceis que é o caso da LA21.

Por outro lado, mantendo o mesmo número de iterações a população grande avalia 5 vezes mais soluções do que a população pequena.

Em média com o aumento do tamanho da população existe um aumento de desempenho entre 0,67% e 2,59%.

Os resultados estão de acordo com o previsto, e vão ser a base de comparação dos testes consequentes.

5.2.1.2 - Evolução das soluções

A **Figura 35** demostra a evolução dos resultados obtidos no AG das instâncias em estudo ao longo das 500 iterações, com a população grande (100 indivíduos).

Como seria de esperar existe uma evolução mais acentuada nas primeiras 100 iterações, (20% do número total de iterações).

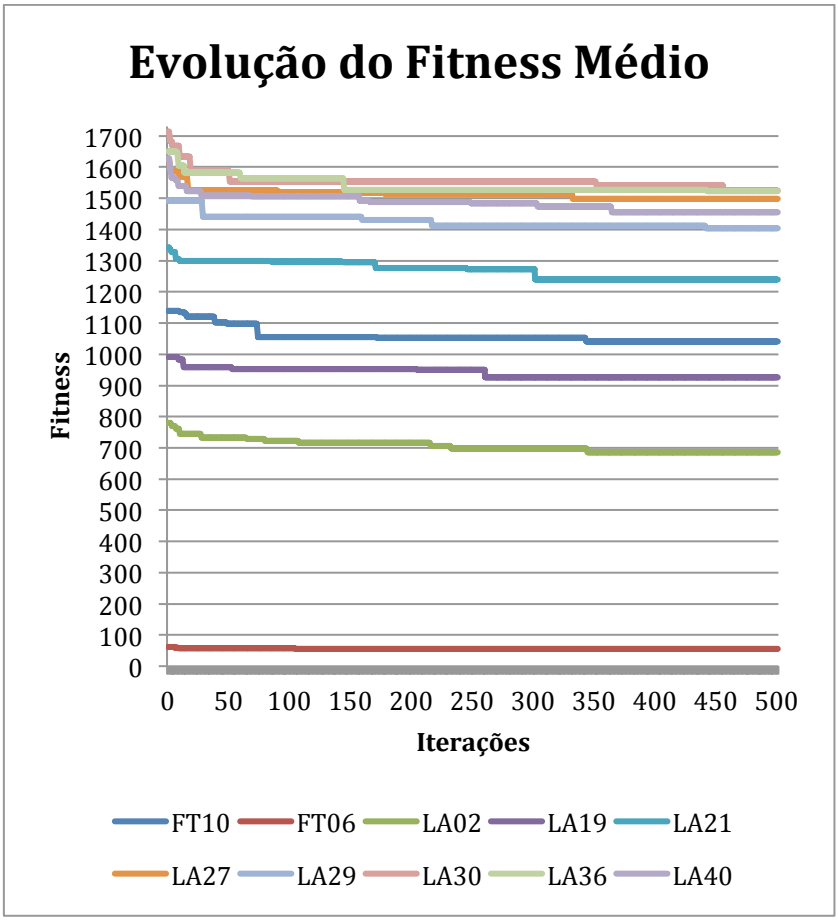


Figura 35 - Evolução dos resultados obtidos ao longo das 500 iterações

5.2.1.3 - Resultados das instâncias FT10 e LA36

Verifica-se uma maior evolução nas primeiras 50 iterações do AG na instância FT10.

Em seguida o algoritmo evolui pouco até as 250 iterações, acentuando-se novamente a melhoria da solução após as 300 iterações.

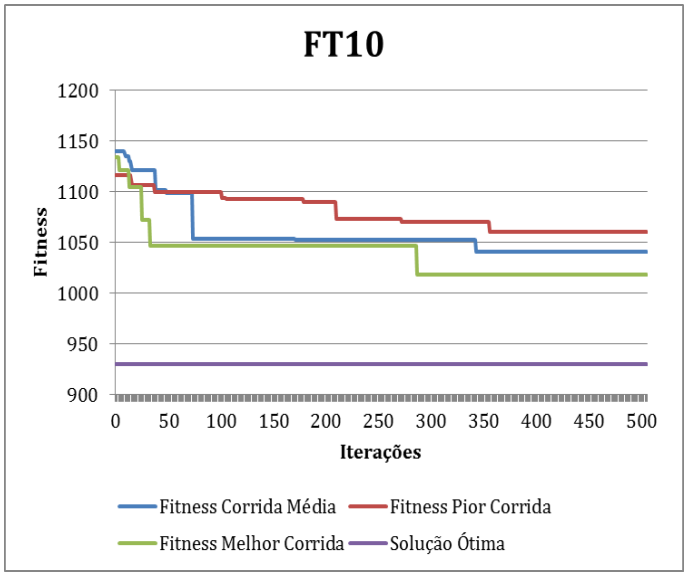


Figura 36 - Evolução dos resultados obtidos na instância FT10

Na instância LA36, o AG tem um comportamento ligeiramente diferente da anterior.

Há uma acentuada evolução nas primeiras 20 iterações, e evolui de forma constante até às 450 iterações. É de esperar que com o aumento de iterações o AG consiga obter soluções de melhor qualidade.

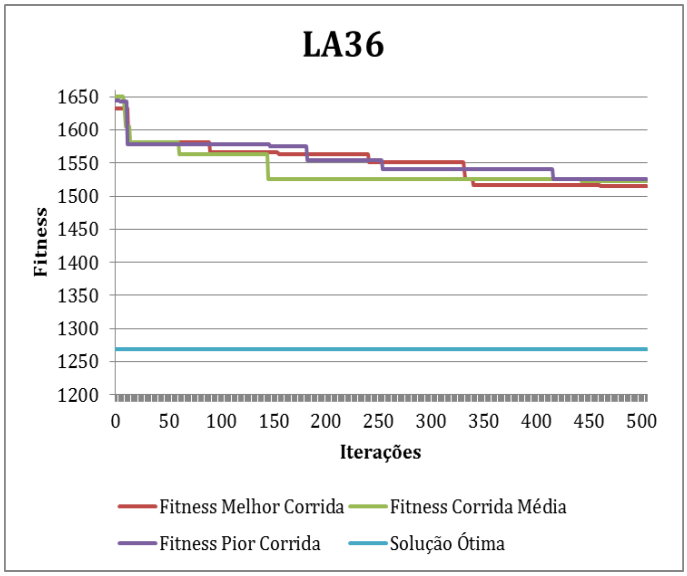


Figura 37 - Evolução dos resultados obtidos na instância LA36.

5.2.2 - P2

Comparação dos resultados obtidos com diferentes taxas de cruzamento

Nesta experiência são obtidos resultados diminuindo a taxa de cruzamento utilizada na experiência anterior. A configuração utilizada para o AG é a seguinte:

Tamanho da População	**100**
Taxa de Cruzamento	**0,25**
Elite	**5**
Indivíduos cruzados (%)	**60**
Indivíduos cruzados e mutados (%)	**20**
Indivíduos mutados (%)	**20**
Mérito	**Min. Makespan**
Critério de Paragem	**500 iterações**

Configuração 2 - Diminuição da taxas de cruzamento

A escolha de uma correta taxa de cruzamento é crítica para a evolução do AG. Todavia não existe uma taxa de cruzamento mais adequada para todas as instâncias, pois varia conforme o problema a resolver.

No AG deve-se encontrar um equilíbrio entre a *exploração* de novas soluções e no *desenvolvimento* da capacidade de optimização do algoritmo.

A propriedade de exploração, procura encontrar novas soluções em todo o espaço de busca. Por outro lado o *desenvolvimento* da solução, concentra-se num dado ponto, que pode ser um mínimo local ou na melhor das hipóteses, um mínimo global.

São os operadores genéticos de cruzamento e mutação, os principais responsáveis por estas propriedades dos AG.

É mais vantajoso para o AG que haja uma maior procura do espaço de soluções (*exploração*) no início do processo de pesquisa, para assegurar a cobertura e diversidade da população. Por outro lado deve-se intensificar o *desenvolvimento* da solução no final do processo de pesquisa, para assegurar a convergência da população para um óptimo global.

Quando o AG converge para um óptimo local, se possível deve-se aumentar a diversidade da população, para aumentar o espaço de procura.

Em muitos casos, para uma procura de solução que cubra um maior espaço de soluções são utilizadas taxas de cruzamento não inferiores a 0,6.

Esta taxa permite que entrem novos indivíduos para a nova população ao longo das iterações.

Deste modo há maior diversidade genética na população ao longo das iterações, permitindo a procura num maior espaço de soluções e assim evitar que o AG possa convergir mais rapidamente.

Algoritmos genéticos com taxas de cruzamento reduzidas convergem mais rapidamente para um mínimo. Todavia há uma maior probabilidade de uma convergência prematura para um mínimo local.

Na **Tabela 16**, são demonstrados os resultados da corrida média obtidos através da redução da taxa de cruzamento:

Tabela 16 - Resultados com a Configuração 2

Instância	Taxa de Cruzamento (Tc)	Solução Ótima (opt)	Corrida Média	Diferença de Tc	Ganho Relativo (%)
FT06	0,65	55	55		
	0,25	55	56	-1	1,82
FT10	0,65	930	1041		
	0,25	930	1004	37	-3,55
LA02	0,65	655	679		
	0,25	655	645	34	-5,01
LA19	0,65	842	926		
	0,25	842	884	42	-4,54
LA21	0,65	1046	1239		
	0,25	1046	1195	44	-3,55
LA27	0,65	1235	1498		
	0,25	1235	1431	67	-4,47
LA29	0,65	1152	1403		
	0,25	1152	1346	57	-4,06
LA30	0,65	1355	1525		
	0,25	1355	1506	19	-1,25
LA36	0,65	1268	1526		
	0,25	1268	1490	36	-2,36
LA40	0,65	1222	1455		
	0,25	1222	1429	26	-1,79

5.2.2.1 - Análise de resultados

Verifica-se uma melhoria nos resultados obtidos na maior parte das instâncias, com a diminuição da taxa de cruzamento.

Este resultado é devido ao aumento do *desenvolvimento* da solução, fazendo com que o algoritmo convirja para um mínimo local. Esta diminuição da taxa de cruzamento teve influência para a melhoria dos resultados finais do AG. Contudo para se obter melhores resultados poder-se-ia adoptar técnicas de injeção de novos

indivíduos com diversidade genética diferente quando o AG não conseguisse melhorar ao longo de X iterações. Desta forma evitava-se a convergência prematura.

Esta conclusão pode ser diferente para um maior número de iterações, onde poderá ser evidente que a convergência prematura facilita ter bons resultados em pouco tempo/iterações, por outro lado não terá capacidade de evoluir em mais iterações. Leva-nos a concluir da necessidade do tamanho de população e taxa de cruzamento deverem ser afinados em conjunto.

5.2.2.2 - Resultados das instâncias FT10 e LA36

A instância FT10 tem uma rápida evolução nas primeiras 150 iterações.

Contudo existiram 200 iterações em que o AG não conseguiu melhorar a solução (ou teve pouca melhoria).

Na parte final do processo verificou-se uma melhoria, podendo ser explicada com a aplicação de outros operadores genéticos (*mutação*).

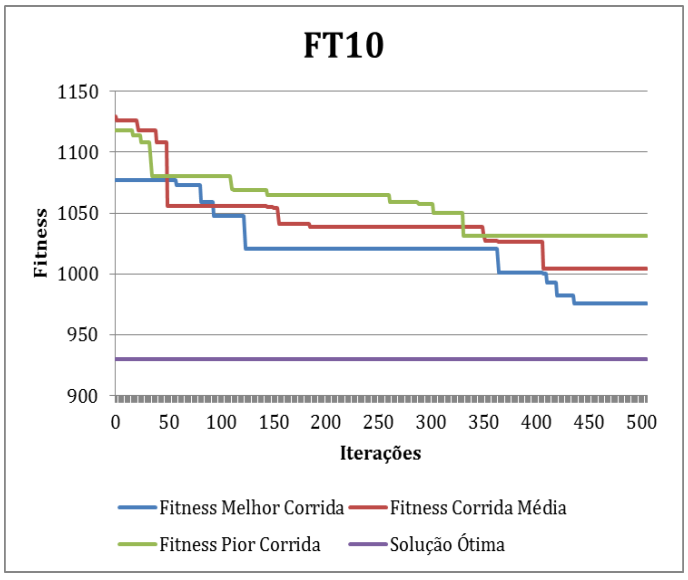


Figura 38 - Evolução dos resultados obtidos na instância FT10

A instância LA36 teve uma evolução idêntica, pelo que houve apenas uma melhoria substancial nas primeiras 150 iterações.

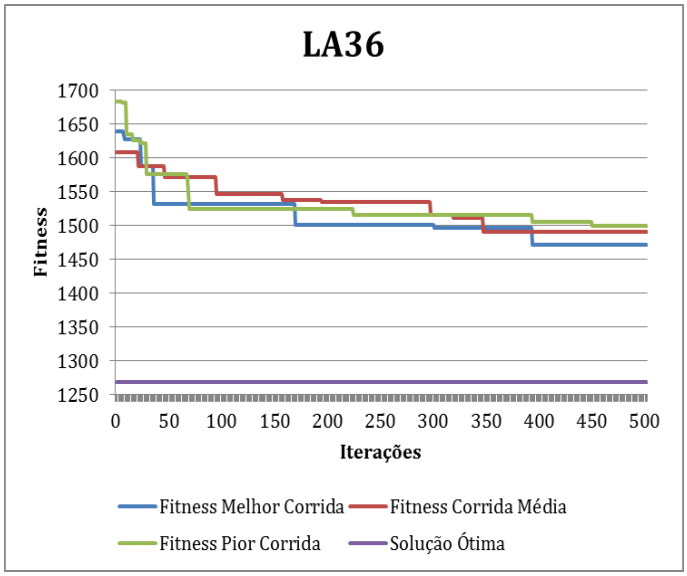


Figura 39 - Evolução dos resultados obtidos na instância LA36.

5.2.2.3 - Comparação de resultados das instâncias FT10 e LA36 nas experiências P1 e P2

Ao longo das 500 iterações verifica-se que há uma melhoria constante na evolução das soluções em P2. Deve-se ao facto do AG se concentrar na melhoria de um mínimo encontrado do espaço de soluções.

Em P1 a melhoria é mais localizada em vários pontos do espaço de soluções, sendo mais lenta a melhoria de uma boa solução.

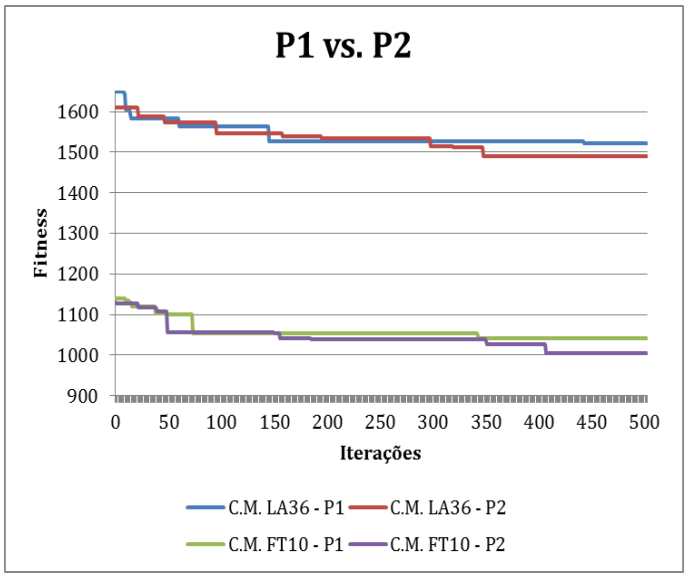


Figura 40 - Resultados obtidos com taxa de cruzamento P1 vs. P2.

5.2.3 – P3

Comparação dos resultados obtidos com o aumento de % de população cruzada

Nesta experiência são obtidos resultados aumentando a incidência do cruzamento sobre a totalidade da população e diminuindo o número de indivíduos mutados.

Com o aumento de população cruzada o AG aumenta a fase de *desenvolvimento* e optimização de solução.

Com a diminuição do número de indivíduos mutados diminui a diversidade genética da população o que poderá levar a uma convergência prematura para mínimos locais.

Em instâncias mais simples é esperado que não haja uma melhoria significativa de soluções pois o AG não necessita de uma população geneticamente muito variada para encontrar boas soluções.

Em instâncias mais complexas o AG irá explorar valores vizinhos de boas soluções já encontradas, havendo a possibilidade de evoluir para mínimos locais.

A configuração utilizada para o AG é a seguinte:

Tamanho da População	**100**
Taxa de Cruzamento	**0,65**
Elite	**5%**
Indivíduos cruzados (%)	**80**
Indivíduos cruzados e mutados (%)	**0**
Individuos mutados (%)	**20**
Mérito	**Min. Makespan**
Critério de Paragem	**500 iterações**

Configuração 3 - Aumento da % de população cruzada

A **Tabela 17** ilustra os valores obtidos com as diferentes configurações do número total de indivíduos cruzados e mutados. É realizada uma comparação e ganho entre as duas configurações:

Tabela 17 - Resultados com a Configuração 3

Instância	*Proporção crossover (Pc)*	*Solução Ótima (opt)*	*Corrida Média*	*Diferença de Pc*	*Ganho Relativo (%)*
FT06	0,6 - 0,2 - 0,2	55	55		
	0,8 - 0,0 - 02	55	55	0	0,00
FT10	0,6 - 0,2 - 0,2	930	1041		
	0,8 - 0,0 - 02	930	1030	11	-1,06
LA02	0,6 - 0,2 - 0,2	655	679		
	0,8 - 0,0 - 02	655	668	11	-1,62
LA19	0,6 - 0,2 - 0,2	842	926		
	0,8 - 0,0 - 02	842	915	11	-1,19
LA21	0,6 - 0,2 - 0,2	1046	1239		
	0,8 - 0,0 - 02	1046	1223	16	-1,29
LA27	0,6 - 0,2 - 0,2	1235	1498		
	0,8 - 0,0 - 02	1235	1480	18	-1,20
LA29	0,6 - 0,2 - 0,2	1152	1403		
	0,8 - 0,0 - 02	1152	1384	19	-1,35
LA30	0,6 - 0,2 - 0,2	1355	1525		
	0,8 - 0,0 - 02	1355	1533	-8	0,52
LA36	0,6 - 0,2 - 0,2	1268	1526		
	0,8 - 0,0 - 02	1268	1513	13	-0,85
LA40	0,6 - 0,2 - 0,2	1222	1455		
	0,8 - 0,0 - 02	1222	1445	10	-0,69

5.2.3.1 - Análise de resultados

Como seria de esperar, na maior parte das instâncias verifica-se uma melhoria de soluções.

Para esta configuração do AG, a falta de diversidade genética da população devido à diminuição da população mutada, não tem um efeito negativo no

desenvolvimento do algoritmo. Deve-se ao facto de não existirem iterações suficientes para que a fase de exploração do espaço de soluções seja afectada.

Apenas no caso da instância LA30 se obtiveram resultados piores. Esta instância é considerada um problema fácil de grande dimensão pelo que a falta de diversidade "atrasou" ou impediu obter o mínimo global. Na maioria dos problemas o AG encontrou melhores resultados, verificando-se melhorias entre 0,69% e 1,62%.

No problema FT06 ambas as configurações chegaram à solução óptima devido a esta ser uma instância considerada simples.

5.2.3.2 - Resultados das instâncias FT10 e LA36

Nas primeiras 200 iterações do AG no problema FT10 regista-se a maior evolução das soluções encontradas. Contudo apenas a partir da iteração 450 é que se verificam novas melhorias significativas. É entre a iteração 200 e 450 que deveria haver entrada de novos indivíduos geneticamente diferentes da população atual e assim permitir a pesquisa em diferentes pontos do espaço de soluções.

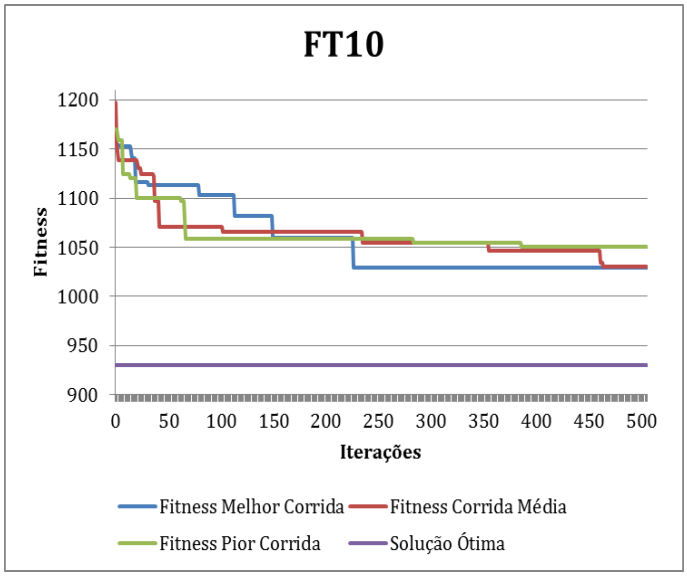


Figura 41 - Evolução dos resultados obtidos na instância FT10

No problema LA36 existe uma evolução constante ao longo das 500 iterações.

Apenas com mais iterações do AG se poderia concluir que a falta de diversidade genética iria influenciar a evolução do algoritmo.

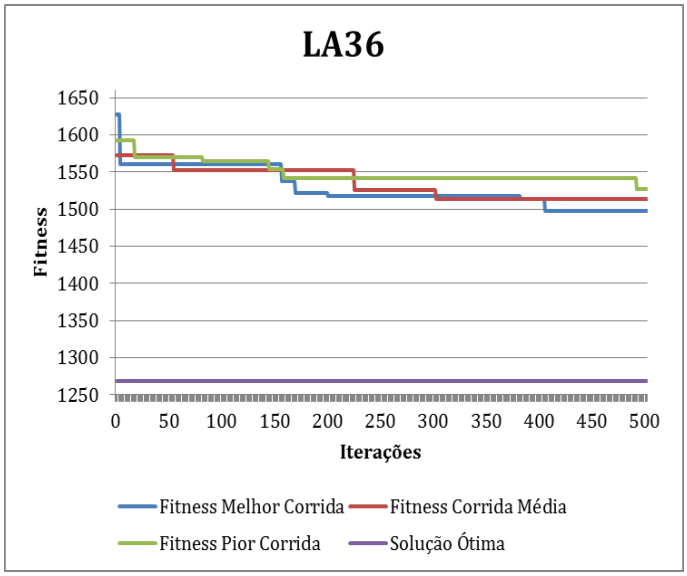


Figura 42 - Evolução dos resultados obtidos na instância LA36.

5.2.3.3 - Comparação de resultados das instâncias FT10 e LA36 nas experiências P1 e P3

Através da comparação dos resultados obtidos nas duas experiências (*P1 e P3*) verifica-se uma ligeira melhoria de soluções. Deve-se ao facto de se ter aumentado a propriedade de *desenvolvimento* e otimização das soluções encontradas.

Esta medida é elitista devido a existir pouca possibilidade de entrada de novos indivíduos geneticamente diferentes ao longo das iterações. O AG atribui mais importância aos melhores indivíduos da população ao longo das iterações.

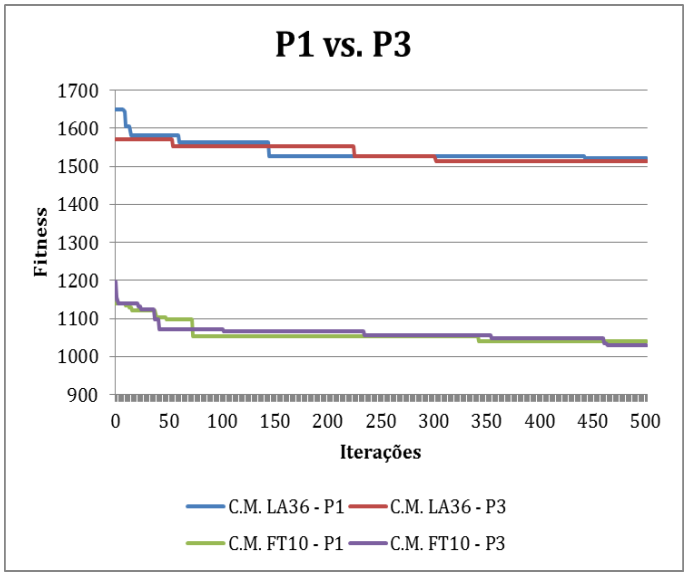


Figura 43 - Resultados obtidos com o aumento de % de população cruzada

5.2.4 – P4

Comparação dos resultados obtidos com o aumento de iterações.

Nesta experiência são obtidos resultados aumentando o número de iterações do AG onde se espera que haja um acentuado melhoramento da solução obtida.

Contudo existe um custo computacional associado a este aumento de iterações.

Deverá existir um equilíbrio entre o número de iterações e o tamanho da população. Teoricamente, um AG necessita de menos iterações quanto maior o número de indivíduos da população. Contudo estes dois parâmetros são característicos de cada problema, não existindo um valor padrão para eles.

Pelos resultados que se obtiveram nas experiências anteriores, verifica-se que existe ainda uma grande margem de evolução dos resultados.

É esperado que o aumento de iterações resulte numa melhoria substancial das soluções encontradas.

A configuração utilizada para o AG é a seguinte:

Tamanho da População	**100**
Taxa de Cruzamento	**0,65**
Elite	**5%**
Indivíduos cruzados (%)	**80**
Indivíduos cruzados e mutados (%)	**0**
Individuos mutados (%)	**20**
Mérito	**Min. Makespan**
Critério de Paragem	**5000 iterações**

Configuração 4 - Aumento do nº de iterações

Esta configuração apenas se diferencia da configuração utilizada em *P1* no número de iterações. Os resultados obtidos são descritos na **Tabela 18**:

Tabela 18 - Resultados com a Configuração 4

Instância	***Iterações (It.)***	***Solução Ótima (opt)***	***Corrida Média***	***Diferença Absoluta It.***	***Ganho Relativo It. (%)***
FT06	500	55	55		
	5000	55	55	0	0,00
FT10	500	930	1041		
	5000	930	981	60	6,12
LA02	500	655	679		
	5000	655	659	20	3,03
LA19	500	842	926		
	5000	842	881	45	5,11
LA21	500	1046	1239		
	5000	1046	1175	64	5,45
LA27	500	1235	1498		
	5000	1235	1441	57	3,96
LA29	500	1152	1403		
	5000	1152	1343	60	4,47
LA30	500	1355	1525		
	5000	1355	1474	51	3,46
LA36	500	1268	1526		
	5000	1268	1442	84	5,83
LA40	500	1222	1455		
	5000	1222	1409	46	3,26

5.2.4.1 - Análise de resultados

Na instância FT06 o AG atingiu a solução óptima com as 500 iterações, pelo que é indiferente o aumento de iterações neste caso.

Nos restantes problemas houve uma melhoria substancial das soluções, em que se apresentaram ganhos na ordem de 3,03% a 6,12%.

Para o AG implementado, o número de iterações mostra-se fundamental na procura de boas soluções. Todavia houve um aumento de tempo e recursos computacionais utilizados.

5.2.4.2 - Resultados das instâncias FT10 e LA36

Na instância FT10, as melhorias substanciais verificam-se nas primeiras 1500 iterações. Contudo o AG continua a melhorar até às 5000 iterações mas de uma forma menos acentuada, que é um comportamento típico nos métodos heurísticos.

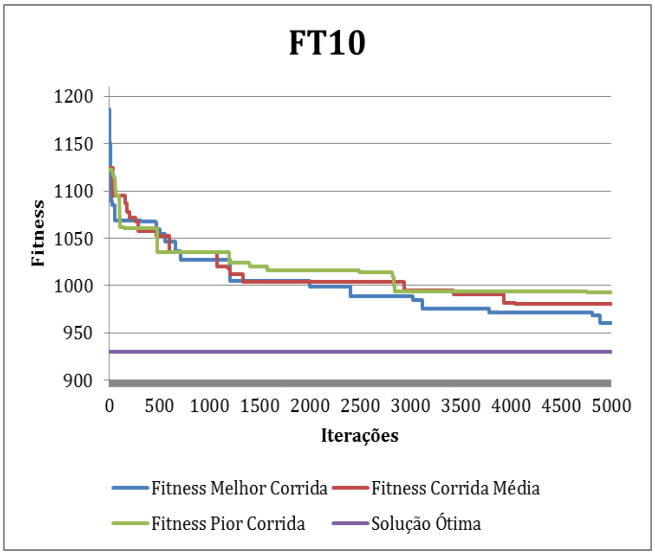


Figura 44 - Evolução dos resultados obtidos na instância LA36

Na instância LA36, as maiores melhorias encontram-se nas primeiras 1000 iterações. Em seguida há um abrandamento na melhoria devido a ter encontrado um mínimo local. A este ponto seria vantajoso uma entrada de indivíduos geneticamente diferentes na população com o objectivo do AG encontrar outro mínimo de melhor valor no espaço de solução, idealmente o mínimo global.

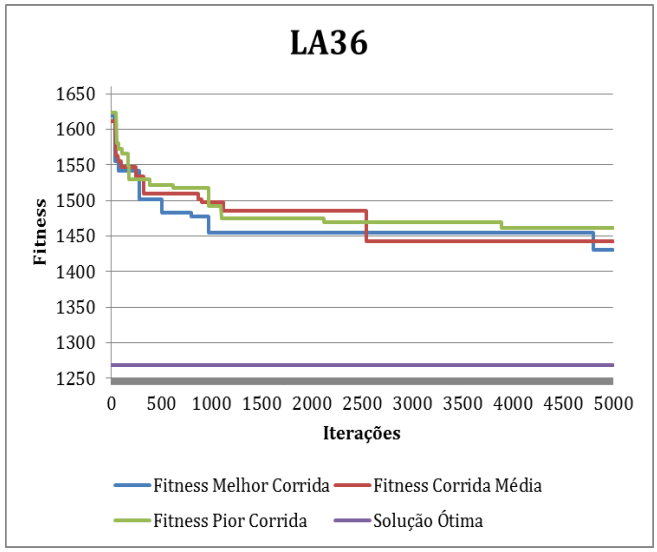


Figura 45 - Evolução dos resultados obtidos na instância LA36

5.2.5- P5

Comparação dos resultados obtidos com recurso a Simulated Annealing (SA)

Nesta experiência são obtidos resultados variando a proporção de iterações com recurso à hibridização do AG com o método de Simulated Annealing (SA).

A substituição de um cromossoma descendente com melhor fitness que o seu progenitor apenas será efectuada de acordo com uma probabilidade. Esta probabilidade tem por base o valor do fitness do cromossoma descendente e a temperatura de arrefecimento do SA. Esta relação é inversamente proporcional, quanto menor for a temperatura, maior será a probabilidade de substituição do descendente pelo progenitor.

A temperatura de arrefecimento vai diminuindo, e é atualizada ao longo do algoritmo até que atinja um dado número de iterações, ficando com o valor 0 quando é atingido esse limite. Nesta fase todos os cromossomas descendentes com melhor fitness que os progenitores passarão para a nova população.

O objectivo será a de impedir a convergência prematura do AG, mantendo uma variedade genética suficiente que possa abranger um maior espaço de soluções.

Para esta experiência estabeleceram-se 3 diferentes limites de atuação do SA.

O SA será desligado e a temperatura de arrefecimento definida a 0 após as:

- 1000 iterações em 5000;
- 2500 iterações em 5000;
- 4000 iterações em 5000;

A configuração utilizada para o AG é a seguinte:

Tamanho da População	100
Taxa de Cruzamento	0,65
Elite	5%
Indivíduos cruzados (%)	80
Indivíduos cruzados e mutados (%)	0
Indivíduos mutados (%)	20
Mérito	Min. Makespan
Critério de Paragem	5000 iterações

Configuração 5 - Variação de iterações com Simulated Annealing

Na **Tabela 19** são comparados os valores obtidos das instâncias FT10 e LA36 com e sem recurso a SA.

Tabela 19 - Resultados com a Configuração 5

Instância	Simulated Annealing	Solução Ótima (opt)	Corrida Média	Ganho S.A. (%)
FT10	Desligado	930	981	-
	2500/2500	930	951	3,15
	1000/4000	930	962	1,98
	4000/1000	930	938	4,58
LA36	Desligado	1268	1442	-
	2500/2500	1268	1435	0,49
	1000/4000	1268	1413	2,05
	4000/1000	1268	1425	1,19

5.2.5.1 - Análise de resultados

A aplicação deste método permitiu uma melhoria das soluções encontradas sem recurso a SA.

Estes resultados explicam-se pelo facto de haver uma maior exploração do espaço de soluções.

Em instâncias diferentes a iteração em que se anula a temperatura de arrefecimento mais adequada varia.

Para a instância FT10 obteve-se o melhor resultado desligando o SA na iteração 4000.

Na a instância LA36 os melhores resultados obtiveram-se desligando o SA na iteração 1000.

Seria necessário incluir mais instâncias para se poder ter uma melhor percepção de qual será a melhor configuração.

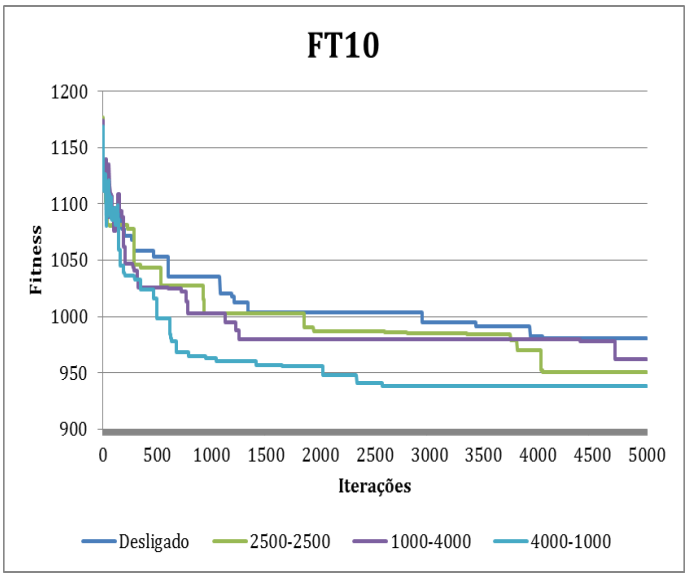


Figura 46 - Evolução dos resultados obtidos na instância FT10

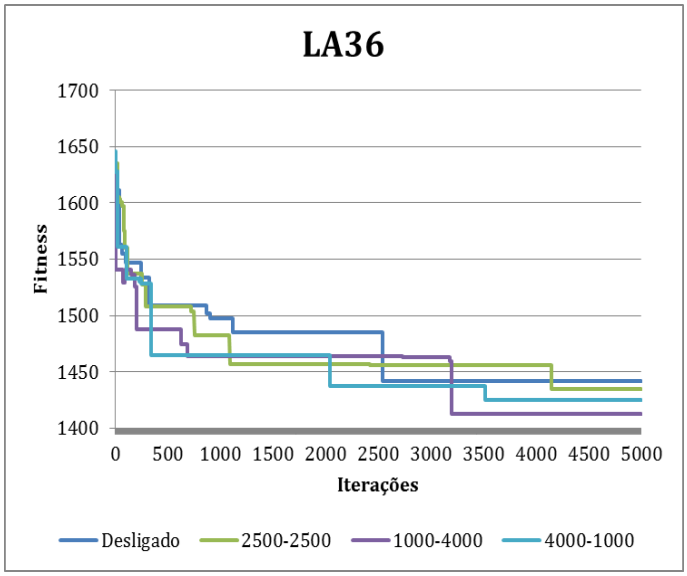


Figura 47 - Evolução dos resultados obtidos na instância LA36

6 – Conclusões e Trabalho Futuro

6.1 - Conclusões

Nesta tese apresenta-se um estudo sobre problemas de planeamento de operações do tipo job shop. Devido à sua natureza combinatória, o planeamento de operações do tipo job shop pertence à classe de problemas *NP-difícil*. Dada a complexidade deste tipo de problemas, torna-se impossível testar todas as soluções possíveis pois tal não seria exequível em tempo computacional útil.

A abordagem ao problema pretende a geração de planos válidos através da implementação das duas seguintes metodologias:

- **Algoritmo Construtivo:**
  - *Algoritmo Giffler e Thomson:* permite gerar todos os planos ativos para o problema JSSP, com *n* ordens de fabrico a serem processadas em *m* máquinas.

- **Meta-Heurísticas:**
  - *Algoritmo Genético:* poderosa ferramenta de pesquisa de melhoramento de soluções, através de métodos evolutivos.
  - *Simulated Annealing:* método de pesquisa e exploração do espaço de soluções, com objetivo de fuga a mínimos locais.

O problema sequenciamento das operações JSSP foi abordado utilizando-se o algoritmo proposto por Giffler e Thompson (GF) , que produz planos ativas. O plano ativo é aquele onde nenhuma operação pode iniciar mais cedo sem atrasar outra operação ou sem violar as restrições de precedência. No final do algoritmo GF, obtém-se uma classificação do plano válido obtido através da minimização do tempo total das operações (makespan).

Para a pesquisa e melhoramento de soluções, implementou-se a hibridização de um Algoritmo Genético com um procedimento de fuga a mínimos locais através do Simulated Annealing.

Com vista a comprovar o desempenho e eficácia dos algoritmos implementados, foram realizados vários testes computacionais. As experiências foram realizadas com base numa seleção de problemas padrão conhecidos na literatura. A seleção dos problemas pretendeu abranger instâncias de diferente níveis de tamanho e dificuldade, para analisar o comportamento dos algoritmos em diferentes ambientes. O primeiro grupo é composto por duas instâncias *(FT06, FT10)* desenvolvidas por Fisher e Thompson (1963). O segundo grupo é composto por oito instâncias *(LA02, LA19, LA21, LA27, LA29, LA30, LA36, LA40)* desenvolvidas por *Lawrence (1984)*.

Para cada instância foram realizadas 10 corridas. Analisou-se os resultados da Melhor Corrida, Corrida Média e Pior Corrida.

Nas experiências realizadas pretendeu-se avaliar o impacto dos parâmetros do Algoritmo Genético, bem como com qual configuração se obtinham melhores resultados. Analisou-se o impacto do número de iterações, tamanho da população, taxa de cruzamento, proporção de população cruzada e mutada bem como várias configurações da aplicação do Simulated Annealing.

Os resultados dos testes confirmam a boa qualidade dos algoritmos desenvolvidos para a resolução do JSSP, bem como a capacidade dos algoritmos genéticos para abordar este tipo de problemas.

Adicionalmente a implementação do Simulated Annealing aumentou a eficácia do sistema na exploração de um maior número de soluções.

Em suma, a originalidade desta tese pode ser descrita em três aspectos principais:

1. Análise e implementação de um Algoritmo Construtivo para a resolução do problema clássico de job shop.
    - Definição de operações, restrições e planos;
    - Implementação do algoritmo de Giffler e Thomson;
    - Sequenciamento de planos válidos;

2. Desenvolvimento do Algoritmo Genético para optimização e pesquisa de soluções.
    - Implementação de operadores genéticos;
    - Desenvolvimento dos parâmetros do AG para aumentar o desempenho, eficiência e eficácia do algoritmo em diferentes problemas.

3. Desenvolvimento de uma abordagem de aumento de pesquisa do espaço de soluções, o Simulated Annealing.
    - Para dar resposta a convergência prematura do AG e tentativa de pesquisa de mínimos globais.

6.2 - Trabalho Futuro

Podem ser adicionadas algumas características ao Algoritmo Genético para aumentar o seu desempenho, tais como:

- Incorporação de conhecimento específico do problema JSSP, como por exemplo, o caminho crítico do problema para que o AG contenha a informação dos genes críticos a atuar;

- Adicionar diferentes tipos de cruzamento e mutação;

- Adicionar informação ao AG para que automaticamente sejam ajustados os graus de exploração de soluções ou desenvolvimento de soluções encontradas, ao longo das iterações;

- Aumentar o tipo de problemas a resolver pelo AG, tais como o problema de escalonamento de operações job shop com recirculação;

- Adaptar o algoritmo implementado para um sistema de apoio à decisão

Existem diferentes tipos de problemas de escalonamento, tais como o Resource Constrained Project Scheduling Problem (RCPSP), Problema de Gestão de Projetos (PGP), Problema de Dimensionamento de Lotes, que podem ser adaptados e resolvidos para o Algoritmo Genético implementado.

Referências Bibliográficas

Referências Bibliográficas

Aarts, Van Laarhoven, Lenstra and Ulder, (1994). “A computational study of local search algorithms for job shop scheduling”. ORSA, Journal on Computing 6, pp. 118-125.

Adams, Balas, Zawack, (1988). “The shifting bottleneck procedure for Job Shop Scheduling”. Management Science 34, 391-401.

Agin, (1966). “Optimum seeking with branch and bound”. Management Science, Vol.13, pp.176-185.

Akers, (1956). “A Graphical Approach to Production Scheduling Problems,” Operations Research 4, No. 2, April.

Alford, (1945).” Production Handbook”. The Ronald Press Co., New York.

Applegate, and Cook, (1991). “A computational study of the job-shop scheduling problem”. ORSA Journal on Computing, Vol. 3, pp. 149-156.

Ashour, (1967). “A decomposition approach for the machine scheduling problem”. Int. J. Prod. Res., 6, 109-122.

Baker and Su, (1974). “Sequencing with due-dates and early start times to minimize maximum tardiness”. Naval Research Logistics Quarterly 21 (1), 171-176.

Baker, (1974). “Introduction to Sequencing and Scheduling”. John Wiley, New York.

Balas, (1969). “Machine Scheduling via Disjunctive Graphs: An Implicit Enumeration Algorithm”. Operations Research, vol. 17, pp.941-957.

Barker and McMahon, (1985). “Scheduling the general job shop”. Management Science, Vol. 31, pp. 594-598.

Barnes and Chambers, (1995). “Solving the job shop scheduling problem using tabu search”.

Bean, (1994). “Genetics and Random Keys for Sequencing and Optimization”. ORSA Journal on Computing, Vol. 6, pp. 154-160.

Beasley, Bull, and Martin, (1993). “An overview of genetic algorithms: Part 1, fundamentals”. University Computing, 15(2): 58-69.

Bhaskaran, and Pinedo, (1991). “Dispatching. In: Handbook of Industrial Engineering. G. Salvendy (ed.)”. John Wiley, New York. 1991. Pp. 2182-2198.

Blackstone, Phillips and Hogg, (1982). “A state-of-the-art survey of dispatching rules for manufacturing job shop operations”. International Journal of Production Research, 20(1), 27-45.

Blazewicz, Domschke, Pesch, (1996). “The job shop scheduling problem: Conventional and new solution techniques”. European Journal of Operational Research 93, 1-33.

Blazewicz, Eiselt, Finke, Laporte, Weglarz, (1991). “Scheduling Tasks and Vehicles in a Flexible Manufacturing System”. The International Journal of Flexible Manufacturing Systems, 4 , 5-16.

Blazewicz, Pesch, et al. (2000). “Total late work criteria for shop scheduling problems”.

Bowman, (1959). “The schedule - sequencing problem. Operations Research, Vol.7, pp.621-624.

Boyd and Burlingame, (1996). “A Parallel Algorithm for Solving Difficult JobShop Scheduling Problems”. Operations Research Working Paper, Department of Industrial Engineering, Texas A&M University, College Station, Texas pp.77843-3131, USA.

Brooks and White, (1965). “An algorithm for finding optimal or near optimal solutions to the production scheduling problem”.

Brown, Lomnicki (1966). “Some Applications of Branch and Bound Algorithm to the Machine Sequencing Problem”, Operational Research Quarterly, Vol. 17, pp. 173-186.

Brucker and Jurisch, (1993). “A new lower bound for the job-shop scheduling problem”. European Journal of Operational Research 64, 156-167.

Brucker, Jurisch, and Krsmer, (1992). “The job-shop problem and immediate selection”, Working paper, University of Osnabriick.

Brucker, P., (1988). “An efficient algorithm for the job-shop problem with two jobs”.

Bruns, (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.

Carlier, (1982). “One machine problem”, European Journal of Operational Research 11, pp. 42-47.

Cerny, (1985). “Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm". Journal of Optimization Theory and Applications, 45:41-51.

Cheng, Gen, Tsujimura, (1996). “A tutorial survey of job-shop scheduling problems using genetic algorithms - I”. Computers & Industrial Engineering 30, 983-997.

Chu et al. (1992). “A branch and bound algorithm to minimize total tardiness with different release dates”. Naval Research Logistics (NRL) 39 (2), 265-283

Clark and Gantt (1922). “The Gantt chart, a working tool of management”. New York, Ronald Press.

Conway, Maxwell and Miller (1967). “Theory of Scheduling”. Addison-Wesley: Reading, MA

Davis, (1985). “Job-shop scheduling with genetic algorithm”. In: 1st International Conference on Genetic Algorithms and their Applications, pp. 136-140. Lawrence Erlbaum, Pittsburgh, PA, USA.

De Jong, (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 First

International Conference on Parallel Problem Solving from Nature, Dortmund, Germany.

Dell'Amico (1993). "Applying tabu search to the job-shop scheduling problem". Trubian.

Della Croce, Tadei and Volta, (1995). "A Genetic Algorithm for the Job Shop Problem". Computers and Operations Research, Vol. 22(1), pp. 15-24.

Dorndorf, U. e Pesch, E., (1995). Evolution Based Learning in a Job Shop Environment, Computers and Operations Research, Vol. 22, pp. 25-40.

Eberhart, Kennedy, (1995). "A new optimizer using particle swarm theory". Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43.

Elhaddad (2012). "Combined Simulated Annealing and Genetic Algorithm to Solve Optimization Problems". World Academy of Science, Engineering and Technology 68 2012.

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.

Field and Keller, (1998). "Project Management". London: International Thomson Business Press.

Fisher and Thompson, (1963). "Probabilistic learning combinations of local job-shop scheduling rules". Factory Scheduling Conference (Carnegie Institute of Technology).

Fisher et al. (1983). "Surrogate duality relaxation for job-shop scheduling".

Fox and McMahon, (1991). "Genetic Operators for Sequencing Problems". Foundations of Genetic Algorithms, Morgan Kaufmann, pp.284-300.

French, (1982). "Sequencing and Scheduling - An Introduction to the Mathematics of the Job-Shop". Ellis Horwood, John-Wiley & Sons, New York.

Gao, Sun, Gen, (2008). “A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems”. Computers & Operations Research 35, 2892-2907.

Garey and Johnson, (1979). “Computers and Intractability: A Guide to the Theory of NP-Completeness”. A Series of Books in the Mathematical Sciences. San Francisco, Calif.: W. H. Freeman and Co..

Gauthier, (1983). “Programação da Produção: uma Abordagem Utilizando Algoritmos Genéticos”. Tese de Doutorado. UFSC. Florianópolis.

Geman, and Hwang, (1986). “Diffusions for Global Optimization”. SIAM J. of Control and Optimization, 24(5), 1031-1043.

Gere, (1966). “Heuristics in jobshop scheduling,” Management Science, Vol. 13, No. 1, pp. 167-175.

Gidas, (1985). “Nonstationary markov chains and convergence of the annealing algorithm”.

Giffler and Thompson, (1960). “Algorithms for Solving Production Scheduling Problems”. Operations Research, Vol. 8(4), pp. 487-503.

Glover, (1977). “Heuristics for integer programming using surrogate constraints”. Decision Sci. 8:156-166.

Glover, (1986). “Future paths for integer programming and links to artificial Intelligence”. Comput. Open Res. 13:533-549.

Glover, (1989). “Tabu search—Part I”. ORSA J. Comput. 1:190-206.

Glover, (1990). “Tabu search—Part II”. ORSA J. Comput. 2:4-32.

Goldberg, (1984). “Computed-Aided Pipeline Operation Using Genetic Algorithms and Rule Learning”. API Pipeline Cybernetics Symp., Houston, Texas, TX.

Goldberg, (1987). “A note on the disruption due to crossover in a binary-coded genetic algorithm”. (RCGA Report No. 87001), Tuscaloosa: University of Alabama, The Clearinghouse for Genetic Algorithms.

Goldberg, (1989). “Genetic Algorithms in Search, Optimization and Machine Learning”. Addison-Wesley, Reading, MA. (1989).

Gonçalves e Beirão, (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 e Mendes, (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.

Gonçalves, Mendes, Resende, (2005). “A hybrid genetic algorithm for the job shop scheduling problem”. European Journal of Operational Research 167, 77-95 (2005).

Granville, Krivanek and Rasson, (1994). “Simulated annealing: A proof of convergence”. IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (6): 652–656. doi:10.1109/34.295910.

Greenberg, (1968). “A branch Bound Solution to the General Scheduling Problem”. Oper. Res. 16, 353-361

Haupt, (1989). “A survey of priority rule-based scheduling”, Operations-Research-Spektrum, Volume 11, Issue 1, pp 3-16

Hoitomt, Luh, and Pattipati, (1993). “Practical approach to job-shop scheduling problems”. IEEE Trans. Rob. Autom. 9/1, Feb., pp.1-13.

Holland, (1975/1992). “Adaptation in Natural and Artificial Systems”. University of Michigan Press, Ann Arbor.

Holley, Kusuoka and Stroock, (1989). “Asymptotics of the spectral gap with applications to the theory of simulated annealing.” J. Funct. Anal. 83, 333-347.

Ignall, and Schrage, (1965). “Application of the branch-and-bound. technique to some flowshop scheduling problems”, Operations Research, Vol. 13, pp.400-412.

Jackson, (1955). “Scheduling a production line to minimize maximum tardiness”. Los Angeles: University of California.

Jackson, (1957). “Simulation Research on Job-Shop Production”. No. 4, December.

Jain and Meeran, (1999). “A state-of-the-art review of job-shop scheduling techniques”. European Journal of Operations Research 113, 390-434 (1999).

Jenigiri, (1996). “Comparative study of efficiency of genetic algorithms and particle swarm optimization technique to solve permutation problems”.

Jeremiah, Lalchandani and Schrage (1964). “Heuristics Rules Toward Optimal Scheduling”. Department of Industrial Engineering, Cornell University, New York, USA.

Kan, (1976). “Machine scheduling problems”. Martinus Nijhoff, La Haya.

Kennedy, (1997). “Minds and cultures: particle swarm implications”. Socially Intelligent Agents: Papers from the 1997 AAAI Fall Symposium, Menlo Park, CA. pp. 67-72, 1997.

Kimbrel and Sviridenko, (2008). “High-multiplicity cyclic job shop scheduling”. Operations Research Letters 36, 574-578 (2008).

Kirkpatrick, Gelatt and Vecchi, (1983). “Optimization by Simulated Annealing”. Science, New Series, Vol. 220, No. 4598, pp. 671-680.

Kolisch, (1995). “Project Scheduling under Resource Constraints”. Physica, Heidelberg.

Krüger, Sotskov and Werner (1995). “Heuristics for Generalized Shop Scheduling Problem Based on Decomposition”. Int. J. on Production Research. 36 (11), 3013–3033

Laarhoven, Aarts, Lenstra, (1992). “Job shop scheduling by simulated annealing”. Operations Research, 40, 113-125.

Lageweg, Lenstra and Kan, (1977), "Job-shop scheduling by implicit enumeration", Management Science 24, 441-450.

Lian, Gu, Jiao, (2006). “A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan”. Applied Mathematics and Computation 183, 1008-1017.

Liepins and Vose, (1991). “Deceptiveness and genetic algorithm dynamics, in Foundations of Genetic Algorithms”. G. Rawlins (ed.), Morgan Kauffmann.

Lin, Horng, Kao, Chen, Run, Lai, Kuo, (2009). “An efficient job-shop scheduling algorithm based on particle swarm optimization”. Expert Systems with Applications 37, 2629-2636 (2009).

Liu, Hao, Wu, (2008). “A prediction based iterative decomposition algorithm for scheduling large-scale job shops”. Mathematical and Computer Modelling 47, 411--421.

Lomnicki, (1965). “A branch-and-bound algorithm for the exact solution of the threemachine scheduling problem”, Operational Research Quarterly, Vol. 16, pp.89-100.

Lourenço and Zwijnenburg, (1996). “Combining the large-step optimization with tabu-search: Application to the job-shop scheduling problem”. Meta-Heuristics, 219-236

Manne, (1960). “On the job-shop scheduling problem”, Operations Research, Vol. 8, no 2 March-April 1960.

Meredith and Mantel (1995). “Project management a managerial approach”. New York: John Wiley & Sons, Inc..

Metropolis, Rosenbluth, Rosenbluth, Teller and Teller, (1953). “Equation of state calculations by fast computing machines”. Journal of Chemical Physics, 21, 1087–1092.

Mitchell, (1996). “An Introduction to Genetic Algorithms”. Cambridge, MA: MIT.

Morton e Pentico, (1993). “Heuristic Scheduling Systems”, John Wiley & Sons, N.Y..

Nemhauser and Wolsey, (1988). “Integer and combinatorial optimization”. Wiley.

Nicholas, (1990). “Hybrid expert system for construction planning and scheduling”.

Nowicki and Smutnicki, (1996). “A fast taboo search algorithm for the job-shop problem”. Management Science 42, 797-813.

Nowicki and Smutnicki, (1996). “A fast tabu search algorithm for the permutation flow shop problem”. European Journal of Operational Research 91, 160-175.

Nowicki and Smutnicki, (2005). “An advanced tabu search algorithm for the job shop problem”. Journal of Scheduling 8, 145-159.

Oliveira, (2006). “A genetic algorithm with a quasi-local search for the job shop problem with recirculation”. In: Applied Soft Computing Technologies: The Challenge of Complexity, pp. 221-234. Springer, Berlin / Heidelberg.

Oliveira, Dias, Pereira, (2010). “Solving the Job Shop Problem with a random keys genetic algorithm with instance parameters”. Proceedings of 2nd International Conference on Engineering Optimization (EngOpt 2010), Lisbon – Portugal, (CDRom) .

Panwalker and Iskander, (1977). “A survey of scheduling rule”. Operations Research 25, 45–61.

Park, Choi, Kim, (2003). “A hybrid genetic algorithm for the job shop scheduling problems”. Computers & Industrial Engineering 45, 597-613.

Pinedo, (2002). “Scheduling: Theory, algorithms, and systems”, 2nd ed. New Jersey: Prentice-Hall Inc., 2002.

Poots, (1980). “An adaptive branching rule for the permutation flowshop problem”. European Journal of Operational Research, 5, 19-25.

Porter, (1929). “Controlling the manufacture of parts on order and for stock by the Gantt progress chart”. ASME Transactions 51, 105–109.

Reeves, (1991). “A genetic algorithm for flowshop sequencing”.

Reeves, (1993). “A Genetic Algorithm for Flowshop Sequencing”. Coventry Polytechnic Working Paper, Coventry, UK.

Rowe and Jackson, (1956). “Research Problems in Production Routing and Scheduling”. Journal of Industrial Engineering, Vol 7, 116-121.

Roy, Sussmann, (1964). “Les problemes d’ordonnancement avec constraintes disjonctives”. Note DS, No. 9 Bis, SEMA, Paris (1964).

Schmitt, (2001). “Theory of Genetic Algorithms”. Theoretical Computer Science 259: 1–61.

Schroeder, Saraph, Benson, (1989). “An instrument for measuring the critical factors of quality management”. Decision Sciences, 20(4), 810-829.

Smith, (1956). “Various optimizers for single stage production”. Naval Research Logistics Quarterly, Volume 3, pages 59-66, 1956.

Sprecher et al., (1995). “Characterization and generation of a general class of resource constrained project scheduling problem”.

Taillard (1994). “Parallel taboo search techniques for the job shop scheduling problem”. ORSA Journal on Computing, 6:2, 108-117.

Taylor, (1911). “Shop Management”. Harper and Bros., New York.

Vaessens, Aarts, Lenstra, (1996). “Job Shop Scheduling by local search”. INFORMS Journal on Computing 8, 302-317.

Van De Velde (1991). “Machine scheduling and Lagrangian relaxation”. Dissertation, CWl Amsterdam.

Vose, (1993). “Modeling simple genetic algorithms”. In Foundations of Genetic Algorithms 2, L. D. Whitley (ed.), Morgan Kaufmann.

Vose, (1999). “The Simple Genetic Algorithm: Foundations and Theory”. Cambridge, MA: MIT Press.

Vose, and Liepins, (1991). “Punctuated equilibria in genetic search”. Complex Systems 5, 31 - 34.

Yang, Sun, Lee, Qian, Lian, (2008). “Clonal Selection Based Memetic Algorithm for Job Shop Scheduling Problems”. Journal of Bionic Engineering 5, 111-119.

Zhang, Li, Guan, Rao, (2007). “A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem”. Computers & Operations Research 53, 313-320.

Apêndice A: Paper - A Genetic Algorithm for the Job Shop on an ASRS Warehouse

Artigo científico desenvolvido para a conferência ICCSA – The International Conference on Computational Science and Its Applications 2012, Salvador da Bahia, Brasil, publicado por Springer, ISBN 978-3-642-31136-9.

A Genetic Algorithm for the Job Shop on an ASRS Warehouse

José Figueiredo[1], José A. Oliveira[1], Luis Dias[1] and Guilherme Pereira[1]

[1] Centre ALGORITMI, University of Minho,
4700-057 Braga, Portugal

josefilipefig@gmail.com, {zan, lsd, gui}@dps.uminho.pt

Abstract. This paper describes the application of a metaheuristic to a real problem that arises within the domain of loads' dispatch inside an automatic warehouse. The truck load operations on an automated storage and retrieval system warehouse could be modeled as a job shop scheduling problem with recirculation. The genetic algorithm is based on random key representation, that is very easy to implement and it allows the use of conventional genetic operators for combinatorial optimization problems. This genetic algorithm includes specific knowledge of the problem to improve its efficiency. A constructive algorithm based in Giffler-Thompson's algorithm is used to generate non delay plans. The constructive algorithm reads the chromosome and decides which operation is scheduled next. This option increases the efficiency of the genetic algorithm. The algorithm was tested using some instances of the real problem and computational results are presented.

Keywords. Genetic Algorithm, Random Keys, Job Shop, Recirculation, ASRS, Warehouses.

1. Introduction

Automatic storage equipments must be efficient in order to justify the investment they imply and also to provide an alternative to conventional storage systems. The efficiency of an automatic storage system depends, among other factors, on the plan for the loading operations of the trucks. In the AS/RS (Automated Storage and Retrieval System) warehouses, where a large number of truckloads are performed on a daily basis, it is necessary to plan and execute accurately the loading procedures in order to fulfill the delivery deadlines.

This paper describes the application of a metaheuristic to a real problem that arises within the domain of loads' dispatch inside an automatic warehouse. An effective

and efficient genetic algorithm is presented to sequence the pallets' retrieval aiming to maximize the warehouse throughput and fulfill the delivery deadlines.
The paper is organized in the following way: next section describes the automatic warehouse type of operations; the third section presents the model adopted, including some remarks about its application and some extensions to the model are also presented; the forth section is dedicated to the characterization of the solution's methodology adopted; the fifth section presents computational results of the developed algorithm; finally the conclusions about the work are discussed.