UNIVERSIDADE FEDERAL DO CEARÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE MESTRADO EM LOGÍSTICA E PESQUISA OPERACIONAL

FRANCISCO REGIS ABREU GOMES

ALGORITMO GENÉTICO APLICADO AOS PROBLEMA DE SEQÜENCIAMENTO PERMUTACIONAL *FLOWSHOP* SEM E COM RESTRIÇÃO DE ESPERA

FORTALEZA - CE
2008

FRANCISCO REGIS ABREU GOMES

Dissertação apresentada ao Programa de Mestrado em Logística e Pesquisa Operacional da Universidade Federal do Ceará, como requisito parcial para obtenção do título de Mestre em Ciências (M.Sc.) em Logística e Pesquisa Operacional.

Área de concentração: Gestão Logística

Orientador: Prof. José Lassance de Castro Silva, D.Sc.

FORTALEZA - CE

2008

G614a Gomes, Francisco Regis Abreu
Algoritmo genético aplicado aos problemas de sequenciamento permutacional Flowshop sem e com restrição de espera [manuscrito]/ Francisco Regis Abreu Gomes, 2008.
141 f.; il.

Orientador: Prof. Dr. José Lassance de Castro Silva
Área de concentração: Gestão Logística
Dissertação (mestrado) – Universidade federal do Ceará, Pró-Reitoria de Pesquisa e Pós-Graduação, Fortaleza, 2008.

1. Pesquisa operacional. 2. problema de sequenciamento permutacional contínuo Flowshop, I. Silva, José Lassance de Castro (orient.) II. Universidade Federal do Ceará – Curso de Mestrado em Logística e Pesquisa Operacional. III. Título.

CDD 003

FRANCISCO REGIS ABREU GOMES

ALGORITMO GENÉTICO APLICADO AOS PROBLEMA DE SEQÜENCIAMENTO PERMUTACIONAL *FLOWSHOP* SEM E COM RESTRIÇÃO DE ESPERA

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Logística e Pesquisa Operacional em 15 de fevereiro de 2008, da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Logística e Pesquisa Operacional. Área de concentração: Gestão Logística.

Aprovado em 15/02/08

BANCA EXAMINADORA

_____

Prof. José Lassance de Castro Silva, D.Sc. (Orientador)

Universidade Federal do Ceará - UFC

_____

Prof. Nei Yoshihiro Soma, Ph.D.

Instuto de Tecnologia de Aeronáutica - ITA

_____

Prof. Antonio Clécio Fontelles Thomaz, D.Sc.

Universidade Estatual do Ceará – UECE

_____

Prof. João Welliandre Carneiro Alexandre, D.Sc.

Universidade Federal do Ceará - UFC

Aos meus Pais.

AGRADECIMENTOS

A Deus, que para mim é fonte de força para não desistir dos meus sonhos.

A Universidade Federal do Ceará (UFC), onde conclui o curso de Engenharia de Produção Mecânica em 2005, pela possibilidade de prosseguir nos estudos acadêmicos através do programa de mestrado interdisciplinar de Logística e Pesquisa Operacional (GESLOG).

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), por ter financiado meus estudos durante o mestrado.

A meu orientador, o professor José Lassance de Castro Silva, por compartilhar sua experiência, pela sua supervisão, pelas correções substanciais que melhoraram a qualidade da minha dissertação e, principalmente por sua confiança em mim.

A todos os professores do GESLOG, principalmente àqueles com quem tive aula.

A todos os servidores técnico-administrativos da UFC, principalmente àqueles ligados diretamente ao mestrado GESLOG.

A meu grande amigo o Engenheiro Civil e M.Sc. Bruno de Athayde Prata, por ter me incentivado a participar da seleção do mestrado GESLOG.

Ao Engenheiro Civil Magno Gonçalves da Costa, aluno no curso de Mestrado em Engenharia Civil, ênfase em recursos hídricos, pelo seminário sobre Algoritmos Genéticos, ministrado nas férias do meio do ano de 2006, que foi um dos fatos responsável por despertar o meu interesse nessa fantástica técnica.

Ao grande amigo Caio Colares Vasconcelos que foi fundamental durante a minha graduação e mestrado, pelos conselhos profissionais, pessoais e técnicos.

Por fim, mas não menos importante, a minha mãe, Regina de Fátima Abreu Gomes, por ter transmitido aos filhos sua paixão pelo saber.

"*O que você sabe depende de quem você quer ser, o modelo do que você pode ser depende do que você sabe*".

Olavo de Carvalho

RESUMO

Neste trabalho foram tratados dois problemas: o primeiro é denominado *Continuous Permutation Flowshop Scheduling Problem* (CPFSP), que possui a restrição de que nenhuma tarefa pode esperar por processamento entre máquinas consecutivas; o segundo é denominado de *Permutation Flowshop Scheduling Problem* (PFSP), em que a restrição anterior não existe. A metaheurística Algoritmo Genético (AG) tem sido aplicada com sucesso ao PFSP, mas até o momento não foi encontrado na literatura algo que mostre que o AG é um bom método para o CPFSP. O objetivo deste trabalho foi desenvolver um AG eficiente paras esses dois problemas, mas que não precisa utilizar inicialização eficiente e/ou hibridização com outra técnica de busca. O desenvolvimento do AG proposto levou em consideração as características, diversificação e a intensificação, que inspiraram a criação de três procedimentos que melhoraram o desempenho do AG proposto. Foram realizados vários experimentos com as instâncias de Taillard (1993), Reeves (1995) e Heller (1960). Os resultados foram comparados com outros métodos encontrados na literatura. Foram construídos polinômios com a utilização de Interpolação Lagrangeana para determinar o tempo execução do AG proposto. Por fim, o método foi aplicado num problema real. Os resultados mostraram que o AG proposto é o melhor método para o CPFSP e que fica muito próximo do melhor AG encontrado na literatura com inicialização eficiente para o PFSP.

Palavras-Chaves: Problema de Sequenciamento Permutacional *Flowshop*, Problema de Sequenciamento Permutacional Contínuo *Flowshop*, Algoritmo Genético, Diversificação e Intensificação.

ABSTRACT

In this work two problems were solved: the first is Continuous Permutation Flowshop Scheduling Problem (CPFSP) it possesses the constraint that no job can wait for processing among serial machines; the second is Permutation Flowshop Scheduling Problem (PFSP), in that the previous restriction does not exist. The metaheuristic Genetic Algorithm (GA) has been applied with success for solving the PFSP, but up to now it was not found in the literature something that shows that GA is a good method for CPFSP. The objective of this work was to develop an efficient GA for both problems, but that does not need to use an initialization efficient and/or hybridization allied with other search technique. The development of proposed GA took in consideration the characteristics, diversification and the intensification, that inspired the creation of three procedures that further improved the proposed GA. Several experiments were accomplished with the instances of Taillard (1993), Reeves (1995) and Heller (1960). The results were compared with other methods found in the literature. Polynomials were built with Lagrangeana's Interpolation use to determine the time execution of proposed GA. Finally, the method was applied in a real problem. The results showed that proposed GA is the best method for CPFSP and that is very close of best GA found in the literature with efficient initialization for PFSP.

Keywords: *Permutacional Flowshop Scheduling Problem*, *Continuous Permutacional Flowshop Scheduling Problem*, Genetic Algorithm, Diversification and Intensification.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AG	Algoritmo Genético
FV	Fink e Voβ
SP	Scheduling Problem
FSP	Flowshop Scheduling Problem
CPFSP	Continuous Permutation Flowshop Scheduling
PFSP	Permutation Flowshop Scheduling Problem
POCP	Problemas de Otimização Combinatorial Permutacional
TTF	Tempo total de fluxo
PMX	Partially Mapped
SJOX	Similar Job Order Crossover
SBOX	Similar Block Order Crossover
SJ2OX	Similar Job 2-Point Order Crossover
SB2OX	Similar Block 2-Point Order Crossover
NN	Nearest Neighbor
Chins	Cheapest Insertion
SD	Steepest Descent
ISD	Iterated steepest descent
IT	Insertion technique
JSSP	Job Shop Scheduling Problem
TS	Tabu Search
OX	Order Crossover

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO

1.1. Considerações Iniciais

As empresas de manufatura enfrentam a difícil tarefa de determinar a melhor seqüência de processamento de seus produtos em suas máquinas que atenda aos objetivos competitivos do negócio. A Pesquisa Operacional denomina este problema como *Scheduling Problem* (SP), na Literatura, e o define como: dado um conjunto de tarefas e um conjunto de máquinas determinar uma seqüência específica que otimize uma função objetivo. Existem vários tipos de SP, por exemplo, o *single machine scheduling problem*, *multiple machine scheduling problem* e *manpower scheduling problem*. Este trabalho trata do *multiple machine scheduling problem* chamada de *Flowshop Scheduling Problem* (FSP). O primeiro artigo publicado sobre este problema foi de Johnson (1954) que formulou e resolveu o *two-machine flowshop problem*. Segundo Gupta e Stafford Jr. (2006) de 1954 a 2004 mais de 1.200 artigos foram publicados abordando diferentes aspectos do FSP.

O FSP é definido como um fluxo unidirecional de n tarefas em m máquinas, i.e., a ordem de processamento de todas as tarefas nas m máquinas é a mesma. Considerando o caso geral do FSP o número de seqüências possíveis e distintas é igual a $(n!)^m$, mesmo para problemas com n e m pequenos a enumeração completa de todas as soluções possíveis e distintas torna-se impossível.

Neste trabalho foram tratados dois problemas da classe FSP. O primeiro problema é uma simplificação do FSP geral, que assume que a seqüência de operações das tarefas processadas em cada máquina é a mesma, por isso, o número de soluções possíveis é reduzido para $n!$, neste caso o problema é denominado *Permutation Flowshop Scheduling Problem* (PFSP). Uma das suposições necessárias para definir o PFSP é que cada tarefa pode esperar pelo processamento entre máquinas consecutivas, i.e., estoque em processo é permitido. Existem processos produtivos onde a suposição anterior não se aplica, i.e., as tarefas não podem parar o processamento entre máquinas consecutivas e, por isso, precisam ser processados continuamente do início ao fim, isto origina um outro problema chamado de *Continuous Permutation Flowshop Scheduling Problem* (CPFSP). O segundo problema e o principal é o CPFSP, dado a sua importância prática e as poucas pesquisas realizadas sobre ele encontradas na literatura.

Ainda sobre o CPFSP, segundo Hall e Sriskandarajah (1994) existem duas razões para a ocorrência de um ambiente de produção contínua. A primeira razão é a tecnologia de produção empregada, por exemplo, a temperatura ou outra característica de um material requer que cada operação siga imediatamente para a próxima etapa. Essas situações são comuns nas indústrias siderúrgica, química, farmacêutica, alimentícia e plástica. Ambientes de manufatura moderna como *just-in-time*, sistemas flexíveis de manufatura e células robóticas exigem uma complexa coordenação no processo de manufatura. Isto também pode ocorrer em empresas de serviço onde o custo de atendimento do cliente é alto. A segunda razão de ocorrência é a falta de espaço de estocagem intermediária, isto ocorre geralmente em linhas de produção automáticas e em sistemas de estoque gerenciado por *kanbans* (cartões informando a quantidade de produtos a serem produzidos), pois nestes casos o estoque em processo tem uma quantidade fixa limitada.

As aplicações futuras do CPFSP seriam principalmente nas indústrias de manufatura moderna, especialmente por causa da automação do manuseio de materiais, segundo Hall e Sriskandarajah (1994).

Este trabalho indica uma boa contribuição científica para o CPFSP principalmente. Além de outros estudos realizados nas resoluções dos problemas.

A metaheurística Algoritmo Genético (AG) baseada na evolução das espécies, tem sido aplicada com sucesso no PFSP (Chen *et al.* (1995), Reeves (1995), Murata *et al.* (1996) e Ruiz *et al.* (2006)). Ruiz *et al.* (2006) desenvolveram um AG que teve um bom desempenho quando aplicado no PFSP. Alguns AGs foram desenvolvidos e aplicados no CPFSP, tais como Chen *et al.* (1996), Aldowaisan e Allahverdi (2003) e Schuster e Framinan (2003). Os AGs de Chen *et al.* (1996) e Aldowaisan e Allahverdi (2003) foram testados em problemas gerados aleatoriamente o que torna difícil a comparação com outros métodos que usaram dados da OR-Library, somente o AG de Schuster e Framinan (2003) foi testado em instâncias conhecidas. Testar um algoritmo em problemas conhecidos e disponíveis na literatura é uma forma de permitir que o método possa ser comparado com outros métodos. Fink e Voβ (2003) desenvolveram várias heurísticas e alguns métodos baseados nas metaheurísticas *Simulated Anneling* e *Tabu Search* para o CPFSP que foram testados nas instâncias de Taillard (1993). Grabowski e Pempera (2005) desenvolveram um

método baseado na metaheurística *Tabu Search* para o CPFSP que obteve um melhor resultado que o AG de Schuster e Framinan (2003). Até o momento não foi encontrado na literatura um trabalho que mostre que o AG é um bom método para o CPFSP devido a este fato escolhemos atacar o CPFSP desenvolvendo um AG que tivesse um bom desempenho.

A primeira justificativa para a escolha do AG é a possibilidade de mostrar que ele pode ser um bom método de resolução quanto ao uso de recursos computacionais. A segunda justificativa é baseada na hipótese de Silva e Soma (2006) que métodos de resolução exata para problemas da classe FSP geralmente só são aplicados em problemas com $n \leq 20$ e que mesmo assim o tempo computacional ainda é muito alto, por isso a importância de desenvolver métodos que encontrem boas soluções em tempo computacional aceitável. A terceira justificativa é que se trata de uma técnica generalista, i.e., pode ser aplicada em vários problemas necessitando somente de poucas modificações. E finalmente, a quarta justificativa é o fato de que o AG está sendo usado no setor produtivo, onde se constatou que em setembro de 1998 através do site EvoWeb, especializado em notícias relacionadas a computação evolucionária, noticiou que em 1997 uma empresa de manufatura foi comprada por US$ 53 milhões por uma empresa de *software*, o alto valor pago foi justificado pelo interesse em adquirir um programa de computador baseado em AG desenvolvido pela empresa de manufatura para fazer o seqüenciamento das ordens de produção da fábrica (EvoWeb, 2007).

O AG foi criado por John Holland durante as décadas de 1960 e 1970 (Holland, 1975). Segundo Haupt e Haupt (2004), o AG é uma técnica baseada nos princípios da genética e seleção natural das espécies. A técnica é formada por uma população de indivíduos que representam as soluções do problema. Cada indivíduo da população é avaliado segundo sua qualidade em relação aos outros indivíduos da população. Os indivíduos são escolhidos por um procedimento inspirado na seleção natural para passarem por operações genéticas que resultam em descendentes que comporão a nova população. Os estudos mostram que a nova população tem a tendência de ter indivíduos com aptidões melhores que os indivíduos da população anterior. Este processo de gerar novas populações é chamado de geração. O melhor indivíduo da última população associado a uma solução do problema é selecionado como a melhor solução encontrada para o problema.

Verificou-se na literatura, que os AGs usados nos problemas da classe FSP apresentam como principais características: a utilização de uma heurística eficiente para criar a população inicial; e uma etapa de hibridização com outra técnica de busca. A inicialização eficiente reduz o tempo necessário para encontrar boas soluções. A etapa de hibridização é usada para melhorar a qualidade da solução obtida. Com estes dois novos componentes fica difícil determinar o quanto da qualidade da solução obtida se deve as características originais do AG criado por Holland. Sendo assim, este trabalho também tem como objetivo desenvolver um AG eficiente para os problemas da classe FSP, principalmente o CPFSP, que não utilize inicialização eficiente e hibridização. Este objetivo se justifica do ponto de vista teórico porque verificará e analisará se um AG sem inicialização eficiente ou hibridização pode ser competitivo com os AGs que usam estas estratégias. Além disso, um AG com estas características pode ser útil quando as heurísticas disponíveis não forem tão eficientes em termos de qualidade das soluções e o tempo computacional ou a hibridização comprometer o custo computacional.

O desenvolvimento do AG proposto levou em consideração as características que fazem a evolução da qualidade das soluções agirem de forma melhor e por mais tempo, as soluções obtidas seriam boas, mesmo sem inicialização eficiente e hibridização. Para isso foram usados dois princípios para guiar a construção do AG. Mitchell (1998) afirmou que no AG a evolução das soluções depende da variação nas aptidões dos indivíduos da população. Outra característica importante é a intensificação no processo de busca (Silva e Soma, 2001; Grabowski e Pempera, 2005; Dréo *et al.*, 2006). Daí se escolheu a diversificação e a intensificação como características importantes para a qualidade de um AG.

Definida a diversidade e a intensificação como as características que atribuiriam qualidade ao AG desenvolvido, tratou-se de encontrar formas de implementar estas características. Depois do desenvolvimento do primeiro AG, seguindo o modelo tradicional, foram desenvolvidos e testados três procedimentos baseados nos princípios da diversificação e intensificação para melhorar o desempenho do AG. O primeiro procedimento é baseado no princípio da diversificação e consiste em permitir, na etapa da formação da nova população do AG, que indivíduos de aptidão menor, mas com características diferentes de todos os outros indivíduos da população tenham chance de serem escolhidos para a nova população. O segundo procedimento é baseado no princípio da intensificação e consiste em fazer o

melhor indivíduo da população passar por um processo genético com outros indivíduos da população mais vezes que o comum. Por fim, o terceiro procedimento é baseado no princípio da diversificação e consiste em realizar uma perturbação em todos os indivíduos da população depois que um estado de estagnação é identificado. Todos estes procedimentos serão detalhados mais adiante.

O objetivo principal deste trabalho é desenvolver um AG eficiente que não utilize inicialização eficiente e hibridização para resolver os problemas CPFSP e PFSP. O AG desenvolvido foi chamado de rAG para diferenciar dos outros AGs existentes.

Os seguintes objetivos específicos precisam ser realizados para que o objetivo principal seja cumprido:

1. Caracterizar de forma clara o PFSP e o CPFSP;
2. Apresentar os modelos matemáticos e combinatorial do PFSP e do CPFSP;
3. Descrever os AGs mais relevantes encontrados na literatura desenvolvidos para resolver o PFSP;
4. Analisar os principais métodos desenvolvidos para resolver o CPFSP;
5. Desenvolver os novos procedimentos que melhorarão a diversidade da população e o processo de intensificação do rAG como forma de o tornar mais eficiente;
6. Realizar experimentos com o rAG para os problemas CPFSP com as instâncias de Taillard (1993), Reeves (1995) e Heller (1960) e para o PFSP com as instâncias de Taillard (1993);
7. Comparar os resultados obtidos pelo rAG com os métodos que foram testados em problemas conhecidos encontrados na revisão bibliográfica para o CPFSP com o critério de desempenho sendo o tempo total de fluxo e o *makespan*;
8. Comparar o rAG com os melhores AG encontrados na revisão bibliográfica para o PFSP com o critério de desempenho sendo o *makespan*;
9. Usar interpolação para construir funções que possam ser usadas para determinar o tempo de execução necessário para o rAG encontrar uma solução de qualidade desejada para o PFSP; e
10. Aplicar o rAG num problema prático analisando seus resultados.

Este trabalho foi dividido em seis capítulos, sendo o Capítulo 1 a introdução. O Capítulo 2 apresenta o PFSP com suas principais características e quatro AGs, os mais relevantes encontrados na literatura aplicados ao problema. O Capítulo 3 trata do CPFSP, onde são apresentados os modelos matemático e combinatorial e os métodos de resolução mais recentes. O Capítulo 4 aborda a técnica AG, onde são descritas as principais características de um AG e do rAG, com suas particularidades. O Capítulo 5 aborda para o rAG: os experimentos com o método, uma aplicação real e o desenvolvimento de funções polinomiais (interpolação) que possam ser usadas para determinar o tempo de execução necessário para o rAG encontrar uma solução de qualidade desejada para o PFSP. O Capítulo 6 apresenta às conclusões e as propostas para futuros trabalhos na área. Por fim, são apresentados a revisão bibliográfica do trabalho e os Anexos I e II, o primeiro anexo apresenta as melhores seqüências de tarefas obtidas pelo rAG para o CPFSP com as instâncias de Taillard (1993) e o segundo as melhores seqüências para as instâncias de Reeves (1995) e Heller (1960).

CAPÍTULO 2 – O PROBLEMA DE SEQÜENCIAMENTO PERMUTACIONAL *FLOWSHOP*

Este capítulo é composto de cinco seções que tratam do PFSP: a primeira seção apresenta a definição do PFSP; a segunda seção apresenta o modelo matemático em Programação Linear Inteiro e Mista do PFSP, com sua complexidade; a terceira seção mostra outra forma de representar o PFSP, baseada na modelagem de Problemas de Otimização Combinatorial Permutacional (POCP); a quarta seção apresenta um histórico da evolução das técnicas aplicadas na solução do PFSP; e finalmente, a quinta seção onde são apresentados os AGs mais relevantes encontrados na literatura aplicados ao PFSP.

2.1. Definição do PFSP

Antes de definir o PFSP é importante esclarecer que *flowshop* não é sinônimo de linha de montagem, mesmo que a característica do *flowshop* seja um fluxo que pareça ser constante de trabalhos através de um conjunto de máquinas em série, conforme Gupta e Stafford Jr. (2006). A seguir são apresentadas três diferenças entre estes dois tipos de modelo de sistema de produção.

a) No ambiente *flowshop* existe uma variedade de produtos e na linha de montagem existe um produto padrão;
b) No ambiente *flowshop* as tarefas não são obrigadas a passarem em todas as máquinas dependendo das necessidades tecnológicas e na linha de montagem todas as tarefas têm que passar por todas as estações de trabalho; e
c) No ambiente *flowshop* cada tarefa tem seu próprio tempo de processamento em cada máquina e na linha de montagem todas as unidades dos produtos têm o mesmo tempo padrão em cada estação de trabalho.

Assim o PFSP pode ser definido como sendo: um conjunto de *n* tarefas J_1, J_2,..., J_n, onde cada tarefa tem para ser processada *m* máquinas M_1, M_2, ..., M_m. Cada tarefa demanda *m* operações, com uma operação representando o tempo de processamento da tarefa por máquina. As tarefas seguem o mesmo fluxo de operações nas máquinas, i.e., para qualquer j =1, 2, ..., *n*, a tarefa J_j deve ser processada primeira na máquina M_1, depois na máquina M_2, e assim por diante até a última máquina, no caso a máquina M_m, conforme mostra a Figura 2.1. Caso a tarefa J_j não utilize todas as máquinas, o seu fluxo continua sendo o mesmo, todavia com o tempo de operação sendo igual a zero. Uma máquina pode processar somente uma operação de cada vez, e iniciada uma operação, ela deve ser processada até a sua conclusão. O número de seqüências distintas possíveis para realização das tarefas nas máquinas é grande, i.e., *O* (*n*!). O problema consiste em realizar todas as *n* tarefas no menor tempo possível (Silva e Soma, 2006).

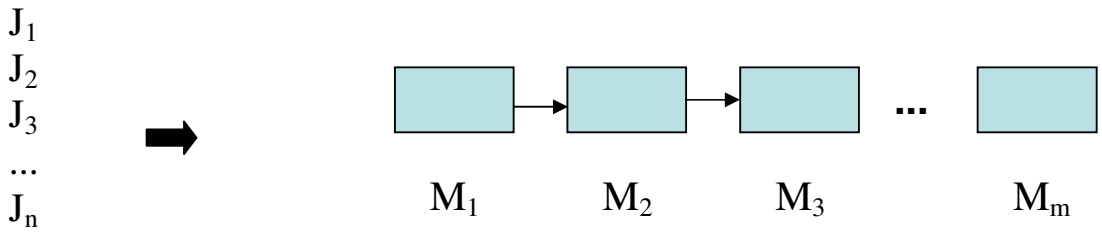


Figura 2.1 – Ilustração do PFSP.

Conforme Silva e Soma (2006) um *input* do PFSP é dado por *n*, *m* e uma matriz P(*n* x *m*) de elementos não negativos, onde P_{ij} denota o tempo de processamento da tarefa J_j na máquina M_i. Seguindo os 4 parâmetros da notação A/B/C/D adotada por Conway *et al.* (1967), o problema é classificado como *n*/m/P/F_{max}. Na menos antiga notação paramétrica α/β/γ, proposta por Graham *et al.* (1979), o problema é denotado como sendo F/prmu/Cmax. O PFSP pertence à classe dos problemas NP-completo, no sentido forte, quando m≥3, conforme Garey e Johnson (1979), no caso em que m=2, o problema pode ser solucionado através de um algoritmo em tempo polinomial.

Suposições relacionadas às tarefas:

J1 – Cada tarefa é liberada para a fábrica no começo do período de programação.
J2 – Cada tarefa pode ter sua própria data de entrega fixa e não sujeita a mudança.
J3 – Cada tarefa é independente das demais.
J4 – Cada tarefa consiste de operações específicas que são realizadas por somente uma máquina.
J5 – Cada tarefa tem uma seqüência tecnológica preestabelecida fixa e que é igual para todas as demais tarefas.
J6 – Cada operação de uma tarefa requer um tempo de processamento finito e conhecido para ser processada nas várias máquinas. Nesse tempo de processamento estão incluídos tempos de transporte, *setup* e outros. O tempo de processamento é independente dos tempos de processamento das tarefas anteriores e posteriores.
J7 – Cada tarefa é processada não mais que uma vez em cada máquina.
J8 **–** Cada tarefa pode esperar entre máquinas consecutivas, ou seja, estoque em processo é permitido.

Suposições em relação às máquinas:

M1 – Cada setor é composto de somente uma máquina e a fábrica tem somente uma máquina de cada tipo.
M2 – Cada máquina está inicialmente desocupada no início do período de programação.
M3 – Cada máquina na fábrica opera independentemente das outras e, por isso, pode operar na taxa de produção máxima.
M4 – Cada máquina só pode processar uma tarefa por vez.
M5 – Cada máquina está continuamente disponível para processar tarefas durante o período de programação e não há interrupções devido a quebras, manutenção ou outras causas.

Suposições relacionadas às políticas de operação

P1 – Cada tarefa é processada tão logo seja possível. Por isso, não há intenção de fazer a tarefa ficar esperando ou fazer a máquina ficar ociosa.

P2 – Cada tarefa é considerada uma entidade individual mesmo que possa ser composta por um conjunto de unidades.

P3 – Cada tarefa uma vez iniciada é processada até o fim, ou seja, o cancelamento de tarefas não é permitido.

P4 – Cada operação de uma tarefa uma vez iniciada numa máquina é completada antes que outra tarefa possa começar na mesma máquina, ou seja, nenhuma preempção é permitida.

P5 – Cada tarefa é processada somente uma vez em cada máquina. Essa suposição é resultado das suposições J5 e P2.

P6 – Cada máquina possui área de estoque suficiente para acomodar as tarefas em espera para serem processadas.

P7 – Cada máquina está completamente alocada às tarefas consideradas durante todo o período de programação, ou seja, as máquinas não são usadas para nenhum outro plano de produção.

P8 – Cada máquina processa as tarefas na mesma ordem.

É importante conhecer estas suposições já que os outros problemas da classe FSP foram criados a partir de alguma modificação nelas. Este é o caso do *sequence dependent setup time Flowshop Scheduling Problem* que foi criado a partir da alteração da suposição *J6* que deixa de considerar o tempo de *setup* como fazendo parte do tempo de processamento e passa a considerar os dois tempos separadamente.

2.2. O Modelo Matemático do PFSP

A seguir são apresentados a notação necessária e o modelo matemático em Programação Linear Inteiro e Mista do PFSP.

Notação:

a) T_{ij} é uma variável do modelo que representa o tempo para iniciar o processamento da tarefa j na máquina i, com $1 \leq j \leq n$ e $1 \leq i \leq m$; e

b) W é um número bastante grande;

c) X_{ijk}, é uma variável binária definida por:

$$X_{ijk} = \begin{cases} 1, \text{se a tarefa } j \text{ precede a tarefa } k \text{ na máquina } i. \\ 0, \text{caso contrário.} \end{cases},$$

com $i = 1, 2, 3, ..., m$, $j = 1, 2, 3,..., (n\text{-}1)$ e $k = (j+1), (j+2),..., n$.

Observações:

i) São dados do problema: n, m e a matriz P (cf. pág. 8);
ii) Em (c), j e k não variam de 1 até n, porque iriam comparar as tarefas j e k na mesma máquina duas vezes; e
iii) As variáveis do tipo X_{ijk} estabelecem a seqüência de processamento das tarefas em cada uma das m máquinas.

$$\text{Minimizar } Z = \sum_{j=1}^{n} T_{mj} \qquad 2.1$$

sujeito a:

$$T_{rj} \geq T_{r-1,j} + p_{r-1,j}\ ,\ \forall\ r = 2,3,...,m \text{ e } j = 1, 2,..., n. \qquad 2.2$$

$$\left.\begin{array}{ll} T_{ik} \geq T_{ij} + p_{ij} - W * (1 - X_{ijk}) & 2.3 \\ T_{ij} \geq T_{ik} + p_{ik} - W * X_{ijk} & 2.4 \\ x_{ijk} \in \{0,1\} & 2.5 \end{array}\right\} \quad \forall\ 1 \leq i \leq m, j = 1, 2,..., n\text{-}1 \text{ e } k = j\text{+}1, j\text{+}2,...n$$

$$T_{ij} \geq 0,\ \forall\ 1 \leq i \leq m \text{ e } 1 \leq j \leq n \qquad 2.6$$

Onde:

a) A função objetivo 2.1 tenta minimizar o tempo para a conclusão das n tarefas nas m máquinas através da obtenção do menor tempo para iniciar cada tarefa na última máquina. Para o caso da tarefa j não utilizar a máquina m substitui-se T_{mj} por T_{rj}, onde r é a última máquina a ser utilizada pela tarefa j com tempo maior de zero;
b) O grupo de restrições 2.2 representa o fluxo que as tarefas devem seguir para serem concluídas. Cada equação do tipo 2.2 determina que a (i+1)-*ésima* operação da tarefa j não pode iniciar até que a *i-ésima* operação da tarefa j na máquina i seja concluída.
c) Os grupos de restrições 2.3 e 2.4 comparam a relação de precedência das n tarefas nas m máquinas. Estes grupos de restrições também não permitem que uma máquina processe duas tarefas ao mesmo tempo;
d) As restrições do grupo 2.5 representam o atendimento à definição da variável X_{ijk} como sendo binária;

e) As restrições do grupo 2.6 definem a não-negatividade dos tempos de processamento; e

f) O valor de W faz com que uma das restrições do grupo 2.3 se torne coerente com a definição da variável X_{ijk} da seguinte forma:

i) Se $X_{ijk} = 1$, então 2.3 fica $T_{ik} \geq T_{ij} + p_{ij}$, coerente com a definição de X_{ijk}, enquanto 2.4 fica $T_{ij} \geq T_{ik} + p_{ik} - W$, ou seja, T_{ij} maior ou igual que um número negativo, logo $T_{ij} \geq T_{ik} + p_{ik} - W$ se torna verdadeiro devido ao grupo de restrição 2.6.

ii) Se $X_{ijk} = 0$, então 2.4 fica $T_{ij} \geq T_{ik} + p_{ik}$, coerente com a definição de X_{ijk}, enquanto 2.3 fica $T_{ik} \geq T_{ij} + p_{ij} - W$, ou seja, T_{ik} maior ou igual que um número negativo, logo $T_{ik} \geq T_{ij} + p_{ij} - W$ se torna verdadeiro devido ao grupo de restrições 2.6.

iii) Podemos adotar W sendo 1000*Max$\{p_{ij}\}$, $\forall\ 1 \leq i \leq m$ e $1 \leq j \leq n$.

Complexidade do modelo:

a) Variáveis do tipo X_{ijk}: m x n x (n - 1) / 2;
b) Variáveis do tipo T_{ij}: m x n;
c) Restrições do grupo 2.2: n x (m - 1);
d) Restrições do grupo 2.3 e 2.4: m x n x (n - 1);
e) Número total de variáveis: m x n (n + 1) / 2; e
f) Número total de restrições: n x (m x n -1).

2.3. O Modelo Combinatorial Permutacional do PFSP

Um POCP pode ser definido por um terno (S, g, n), onde S é o conjunto de todas as soluções do problema, g é sua função ou procedimento que aplica a cada solução viável $s \in S$ um número real e n é uma instância do problema. O número de soluções existentes para um POCP é representado por |S| (cardinalidade de S) e igual a n! (fatorial de n). O objetivo é encontrar uma solução $s^* \in S$ que otimize a um dado critério de desempenho representado pela função g. Representa-se s como uma permutação de n elementos, ou seja, $s=\langle a_1, a_2, ..., a_n \rangle$, com $a_i \neq a_j$, $\forall\ 1 \leq i, j \leq n$ e $i \neq j$.

Segundo Silva e Soma (2006) o PFSP pode ser modelado como um POCP da seguinte forma:

a) Um elemento s= $\langle J_1, J_2, ..., J_n \rangle$ do conjunto de soluções viáveis *S* é representado por uma permutação das *n* tarefas, com a ordem de *s* determinando a seqüência na qual as tarefas serão processadas; e

b) O procedimento *g*, dado a seguir, determina o valor do tempo gasto (t_g) para processar a seqüência *s*, mais precisamente tem-se que t_g é o tempo utilizado no processamento da última tarefa de *s* na última máquina M_m.

Entrada: *m*, *n*, permutação *s*, Matrizes *T*(*m* x *n*)* e *P*(*m* x *n*).
Saída: *g* (tempo gasto para processar todas as *n* tarefas usando a seqüência *s*)

```
for(i=1; i<=m; i++)
        for(j=1; j<=n; j++)  t[i][j]=0;
for(j=1; j<=n; j++) {
        for(i=1; i<=m; i++) {
                if (i==1) {
                if (j>=2) t[1][s[j]]=t[1][s[j-1]]+p[1][s[j-1]];
                } else {
                if (j==1)
                                {t[i][s[1]]=t[i-1][s[1]]+p[i-1][s[1]];
                } else {
                                x=t[i][s[j-1]]+p[i][s[j-1]];
                                y=t[i-1][s[j-1]]+p[i-1][s[j-1]];
                                if (x>=y) t[i][s[j]]=x; else t[i][s[j]]=y;
                }
                }
        }
}
g=t[m][s[n]] + p[m][s[n]];
```

* O elemento t_{ij} representa o tempo para iniciar a tarefa J_j na máquina M_i.

Figura 2.2 – Procedimento para calcular *g* (*s*). Fonte: Silva e Soma (2006).

Para determinar a seqüência *s* com menor valor de *g* seria necessário enumerar e avaliar todas as *n*! seqüências distintas de *S*. A Tabela 2.1 abaixo mostra para alguns valores de *n* a quantidade de soluções distintas de *S* e o tempo computacional, caso o tempo de processamento do procedimento *g* para cada seqüência fosse igual 0,001 segundos. Os resultados da Tabela 2.1 demonstram que para valores de *n* maiores que *20* fica inviabilizada a enumeração completa de todas as soluções.

Tabela 2.1 – Tempos de processamento para valores de n considerando o número de soluções de S.

n	Quant. Soluções	Tempo computacional
5	$1{,}20 \times 10^{2}$	0,12 seg
10	$3{,}63 \times 10^{6}$	3.628,80 seg
20	$2{,}43 \times 10^{18}$	$7{,}71 \times 10^{7}$ anos

2.4. Estado da Arte do PFSP

A análise do histórico do progresso das técnicas utilizadas na resolução dos problemas da classe PFSP serve para situar o método proposto neste trabalho. Gupta e Stafford Jr. (2006) analisaram o desenvolvimento da pesquisa em relação ao PFSP desde o trabalho de Johnson (1954) até 2004. Esse período foi dividido em cinco décadas (1955-1964, 1965-1974, 1974-1984, 1985-1994 e 1995-2004) e para cada período as suposições, as formulações para o problema e as abordagens de solução foram analisadas.

A primeira década tratou o PFSP principalmente do ponto de vista teórico. Além da formulação de Johnson (1954) para duas máquinas foi desenvolvido o *m-machine flowshop* para a minimização do *makespan*. Foram desenvolvidas poucas técnicas para a solução do PFSP. As duas técnicas que mais se destacaram foram à programação matemática (Wagner, 1959; Manne, 1960) e a simulação de Monte Carlo (Sisson, 1959; Muth e Thompson, 1963). O tamanho dos problemas resolvidos eram pequenos por três motivos: i) falta de capacidade computacional; ii) falta de eficientes programas de computador; e iii) a maioria das variações do *two-machine flowshop problem* eram NP-*hard*.

A segunda década apresentou um maior número de técnicas de solução e outras funções objetivo além do *makespan*. Os primeiros a proporem a abordagem combinatorial foram Dudek e Teuton (1964). A técnica *branch and bound* para o PFSP foi desenvolvida por Lomnicki (1965). Nessa época também começou o desenvolvimento das primeiras heurísticas para encontrar boas soluções para o PFSP de grandes dimensões.

Na terceira década com a publicação da teoria do NP-*Completeness* por Garey e Johnson (1979) a pesquisa em relação ao PFSP passou a ter duas direções. Uma direção na tentativa de identificar a complexidade de vários PFSP (Brucker, 1998; Lawler *et al.* 1993) e a outra no desenvolvimento de novas heurísticas. Nessa década também ocorreu a proposição de

vários novos PFSP como o que separa o tempo de *setup* do tempo de processamento, que considera a data de entrega na função objetivo e que considera o tempo de processamento estocástico.

Na quarta década surgiu o *hybrid flowshop* que consiste em cada centro de trabalho poder ser constituído de múltiplas máquinas em paralelo. Nessa década iniciou-se o uso das metaheurísticas (Aarts e Lenstra, 1997): *Tabu Search*; *Simulated Annealing;* e Algoritmo Genético. Também foram desenvolvidas técnicas baseadas em inteligência artificial, sistemas de apoio à decisão e sistemas especialistas (sistemas que utilizam o conhecimento empírico acumulado da resolução de problemas que já ocorreram para ajudar a resolução de novos problemas).

Na quinta década continuou o crescimento na criação de novos problemas, funções objetivo e abordagens de resolução. A principal novidade foi o aumento das pesquisas considerando funções multi-objetivo (T'Kindt e Billaut, 1993).

2.5. Algoritmos Genéticos para a Resolução do PFSP

Os conceitos sobre o AG são apresentados no capítulo 4.

2.5.1. AG de Chen *et al.* (1995)

Chen *et al.* (1995) desenvolveram um AG para o PFSP com o *makespan* como critério de desempenho. O AG foi testado em problemas cujos dados foram extraídos de seqüências de números gerados de maneira pseudo-aleatória. O AG desenvolvido é composto das seguintes partes:

a) Representação dos indivíduos;
b) Geração da população inicial e tamanho da população;
c) Avaliação da aptidão e método de seleção;
d) Operadores genéticos; e
e) Critério de parada.

A representação genética adotada foi a permutacional. Por exemplo, para uma instância com $n = 8$, o indivíduo pode ser representado por qualquer seqüência de oito tarefas como 2 1 7 8 5 3 6 4.

A população inicial é gerada a partir dos métodos CDS desenvolvido por Campbell *et al.* (1970) e de Dannenbring (1977). Os m - 1 primeiros membros da população são gerados pelo método CDS, o elemento de número m é gerado pelo método Dannenbring e do elemento $m + 1$ até o último elemento da população é gerado a partir do primeiro elemento da população, segundo e sucessivamente mediante a troca de posições de duas tarefas escolhidas aleatoriamente.

Segundo *Chen et al.* (1995), depois de vários experimentos que não foram explicitados no artigo, conclui-se que 60 indivíduos era o melhor tamanho para a população.

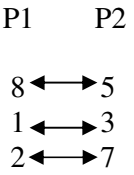
A forma como é calculada a aptidão de cada indivíduo é descrita a seguir. O primeiro passo é calcular o valor do *makespan* de todos os indivíduos da população. O segundo passo é selecionar o $C_{máx}$ que é o *makespan* de maior valor da população. O terceiro passo calcula a aptidão que é igual a diferença entre o valor do *makespan* do indivíduo e o $C_{máx}$. O método de seleção não foi explicitado, a única informação dada foi que a seleção é baseada na aptidão do indivíduo.

O operador de *crossover* utilizado foi o *Partially Mapped* (PMX) desenvolvido por Goldberg (1989). A seguir são apresentados os procedimentos do operador PMX juntamente com uma ilustração para $n = 8$.

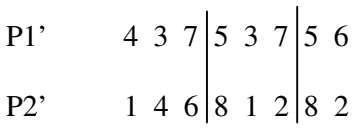
1 – Escolher aleatoriamente um intervalo comum aos dois pais. Por exemplo, as posições de quatro a seis.

P1	4	3	7	8	1	2	5	6
P2	1	4	6	5	3	7	8	2

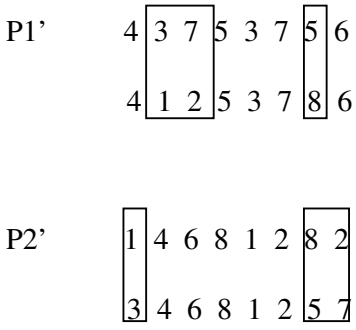
2 – Armazenar relacionadamente os elementos dos dois intervalos selecionados. Na ilustração o armazenamento relacionado é o seguinte:



3 – Trocar os dois intervalos. Na maioria das vezes os indivíduos resultantes não são viáveis porque ocorrem tarefas repetidas. O passo 4 corrige esse problema.



4 – Trocar os elementos repetidos e que não estão dentro dos intervalos selecionados pelas tarefas armazenadas de P1 relacionados às tarefas de P2 e vice-versa. Depois desse passo os indivíduos são totalmente viáveis.



Chen *et al.* (1995) realizaram testes com problemas gerados aleatoriamente para determinar a melhor combinação para as taxas de *crossover* e mutação. Foram usadas para a taxa de *crossover* os valores 1, 0.95 e 0.90 e para a taxa de mutação os valores 0.01, 0.005 e 0. O resultado do experimento mostrou que a melhor combinação foi uma taxa de *crossover* igual a 1 e uma taxa de mutação igual a 0. O significado destes valores é que sempre os indivíduos escolhidos para reprodução passam pelo processo de *crossover* e nunca um indivíduo da população sofre mutação.

O critério de parada do método foi o número de gerações. Após alguns experimentos Chen *et al.* (1995) chegaram à conclusão que depois de 20 gerações a população do AG estagnava e não havia mais melhoria.

2.5.2. AG de Reeves (1995)

Reeves (1995) desenvolveu um AG para o PFSP com o *makespan* sendo o critério de desempenho. O AG tem como principal diferença uma probabilidade de mutação adaptativa e foi testado nas instâncias desenvolvidas pelo próprio autor e de Taillard (1993).

A representação genética utilizada foi a permutacional que é sempre a opção natural para este tipo de problema.

A aptidão de cada indivíduo na população é igual a $v_{máx} - v$, onde $v_{máx}$ é o valor do maior *makespan* da população e v é o valor do *makespan* do indivíduo.

A seleção do método de geração da população inicial deu-se com dois experimentos: i) gerada aleatoriamente; e ii) com um indivíduo gerado pela heurística NEH de Nawaz *et al.* (1983) e o restante da população gerada aleatoriamente. O segundo método obteve soluções tão boas quanto o primeiro método, mas com um tempo computacional menor, por isso, foi o método selecionado.

O método de seleção é composto por dois tipos de seleção. O primeiro pai é selecionado usando o tipo de seleção por *ranking* com probabilidade P_i dada pela Equação 2.6 e o segundo pai é selecionado com probabilidade uniforme de acordo com a aptidão.

$$P_i = \frac{k}{M\,x\,(M + 1)} \qquad 2.6$$

onde:

- i : é um indivíduo da população, $i = 1, 2, ..., M$;
- k : é a posição do indivíduo na população em ordem descendente com relação ao *makespan*; e
- M : é o tamanho da população.

Foram utilizados operadores genéticos de *crossover* e mutação. O operador de *crossover* foi o *one-point crossover* que consiste em escolher um mesmo ponto de corte em cada um dos pais, copiar as tarefas à esquerda do ponto de corte de cada pai para cada um dos descendentes e copiar as tarefas que faltam do outro pai na mesma ordem relativa. A Figura 2.3 ilustra o funcionamento do *one-point crossover*.

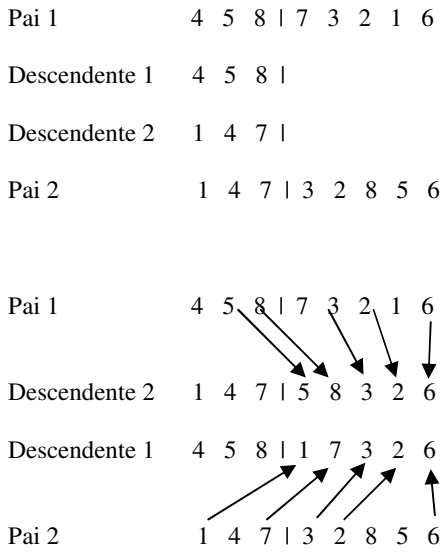


Figura 2.3 – *One-point crossover*.

O operador de mutação utilizado foi o *shift* que consiste em escolher uma tarefa aleatoriamente e colocar numa posição da seqüência escolhida aleatoriamente. Também é utilizada uma estratégia geracional que consiste em inserir os novos indivíduos no lugar dos indivíduos com aptidão menor que a média da aptidão da população. Esta estratégia garante a sobrevivência dos indivíduos com melhor aptidão, mas por outro lado, diminui a diversidade da população.

O critério de parada utilizada foi o tempo de execução, dada a facilidade no momento de realizar os experimentos para comparar com outros métodos.

Durante os experimentos preliminares Reeves (1995) notou que a população convergia prematuramente. Por isso, implementou uma probabilidade de mutação P_m adaptativa. Um parâmetro D é estabelecido para controlar a diversidade da população. A razão v_{min}/v_{med}

que mede a diversidade da população é calculada ao fim de cada geração, onde v_{min} é o valor do menor *makespan* da população e v_{med} é o valor do *makespan* médio da população. Se está razão é maior ou igual a D, a probabilidade de mutação é multiplicada por um fator de decréscimo θ ($0 < \theta < 1$), caso contrário esta probabilidade retorna ao valor inicial P_m^{ini}. A probabilidade P_m^{ini} é alta no início da busca e diminui durante o processo de evolução. Ela retorna a crescer quando a diversidade da população está baixa.

Os valores dos parâmetros usados por Reeves (1995) foram:

- Tamanho da população (M) : 30;
- Probabilidade de *crossover* (P_c) : 1,0;
- Probabilidade inicial de mutação (P_m^{ini}) : 0,8;
- Taxa de decréscimo da probabilidade de mutação (θ) : 0,99; e
- Parâmetro de controle de diversidade (D) : 0,95.

Reeves (1995) também adotou a estratégia de fazer todos os pais passarem pelo processo de *crossover*, já que usou uma taxa de 100%.

2.5.3. AG de Murata *et al.* (1996)

Murata *et al.* (1996) desenvolveram três tipos de estudos com AG para o PFSP com o *makespan* sendo o critério de desempenho. Foram realizados estudos para os operadores genéticos, os valores dos parâmetros e as opções de hibridização.

Neste AG, permutações foram usadas para representar as soluções do problema.

Foram testadas duas formas para calcular a probabilidade de seleção. A Equação 2.7 foi escolhida porque conseguiu a maior pressão de seleção, e assim obteve os melhores resultados. A probabilidade de seleção é representada por P_s, t representa a geração atual, o tamanho da população é representado por N_{pop}, x^i representa um indivíduo e a população atual é representada por $\Psi_t = \{x_t^1, x_t^2, ..., x_t^{Npop}\}$.

$$P_s(x_t^i) = \frac{[f_M(\psi_t) - f(x_t^i)]^2}{\sum_{x_t^i \in \psi_t} [f_M(\psi_t) - f(x_t^i)]^2} \qquad 2.7$$

Onde:

- $f(x_t^i)$ é o valor do *makespan* do indivíduo i, $i = 1, ..., N_{pop}$.
- $f_M(\Psi_t) = \max \{ f(x_t^i) \in \Psi_t \}$.

Quanto aos operadores genéticos de *crossover* e mutação foram testados dez operadores de *crossover* para determinar o melhor para o PFSP. *O two-point crossover* (versão 1) foi o que obteve o melhor desempenho: dois pontos da seqüência são escolhidos aleatoriamente de um dos pais. As tarefas que ficam desses pontos para as extremidades são copiados para o descendente. As tarefas que faltam na seqüência do descendente são copiadas na mesma ordem relativa do outro pai como mostra a Figura 2.4. Foram testados quatro operadores de mutação para determinar o melhor para o PFSP. A mutação *shift* foi a que obteve o melhor desempenho.

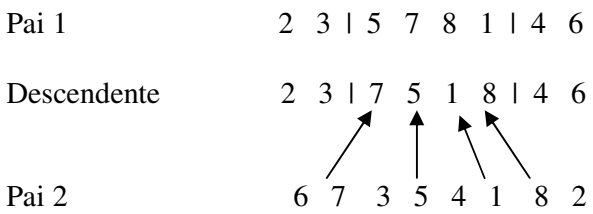


Figura 2.4 – *Two-point crossover* (versão 1).

Estudos também foram realizados para quantificar os parâmetros do AG. A Tabela 2.2 mostra os valores que foram testados. A melhor combinação foi $N_{pop} = 10$, $P_c = 1.0$ e $P_m = 1.0$, tamanho da população, taxa de *crossover* e taxa de mutação, respectivamente. Estes valores significam que todos os indivíduos da população são substituídos por indivíduos gerados no processo de *crossover* e todos os indivíduos da população sofrem mutação. O melhor indivíduo da população anterior é copiado para a nova população no lugar de um indivíduo escolhido aleatoriamente.

Tabela 2.2 – Valores experimentados para os parâmetros do AG de Murata *et al.* (1996). Fonte: Murata *et al.* (1996).

N_{pop}	P_c	P_m
5	0,5	0,5
10	0,6	0,6
20	0,7	0,7
30	0,8	0,8
40	0,9	0,9
50	1,0	1,0

O último estudo realizado foi testar qual o melhor método para hibridizar com o AG. Foram testados duas opções, um algoritmo *simulated annealing* e um algoritmo de busca local. A hibridização do AG com o algoritmo de busca local foi o que obteve o melhor desempenho. Para reduzir o custo computacional da busca local usou-se a estratégia de avaliar só uma parte α da vizinhança de uma solução, por exemplo, α = 10% significa que 10% das soluções da vizinhança são escolhidas aleatoriamente. Não foi mencionada que tipo de estrutura de vizinhança foi utilizada. Para verificar qual seria o melhor valor para o parâmetro α foram feitos testes com os seguintes valores: 100%, 75%, 50%, 10% e 5%. O melhor valor para α foi 75%.

O AG com busca local (α = 75%) é o melhor algoritmo para o PFSP desenvolvido por Murata *et al.* (1996). Ele é composto de sete passos que são descritos a seguir.

1 – Inicialização : gera uma população inicial de indivíduos de forma aleatória de tamanho N_{pop}.

2 – Busca local : aplica a busca local em todos os indivíduos da população se um critério de parada é satisfeito a busca é encerrada senão o processo continua e as novas soluções comporão a população atual. O critério de parada é o seguinte: se depois dos α vizinhos avaliados tiver havido melhoria em algum indivíduo a busca é encerrada, senão, são avaliados todos os vizinhos de todos os indivíduos da população.

3 – Seleção : seleciona N_{pop} pares de pais da população atual de acordo com a probabilidade de seleção.

4 – *Crossover* : Aplica o operador *crossover* a cada par de pais escolhidos com uma probabilidade P_c. Se o operador não for aplicado é escolhido um dos pais para compor a nova população.

5 – Mutação : aplica o operador de mutação a cada indivíduo com a probabilidade P_m, esta probabilidade é referente a cada indivíduo e não a cada tarefa como na representação binária.

6 – Estratégia elitista : adiciona o indivíduo de melhor aptidão da população atual na nova população no lugar de um indivíduo escolhido aleatoriamente.

7 – Critério de parada : finaliza a execução do algoritmo se a condição de parada (tempo de execução) é satisfeita, caso contrário, retorna ao passo 2.

2.5.4. AG de Ruiz *et al.* (2006)

Ruiz *et al.* (2006) desenvolveram dois novos AG para o PFSP sendo também o *makespan* o critério de desempenho. Os AGs têm: inicialização eficiente; estratégia geracional que só aceita indivíduos melhores e com seqüência única; quatro novos operadores de *crossover* que foram desenvolvidos; um procedimento para evitar a convergência prematura; e proposta de uma busca local para a hibridização com o primeiro AG. Foi realizado um projeto de experimento para determinar a melhor combinação de operadores genéticos e valores dos parâmetros dos AGs desenvolvidos.

A representação da solução utilizada foi permutacional, onde a ordem relativa das tarefas na permutação indica a ordem de processamento das mesmas.

Para gerar a população inicial Ruiz *et al.* (2006) desenvolveram uma modificação na heurística NEH de Nawaz *et al.* (1983). A modificação foi a seguinte: depois de ordenar todas as tarefas em ordem decrescente do tempo total de processamento, são escolhidas duas tarefas aleatoriamente e colocadas nas duas primeiras posições, depois disso o procedimento continua igual ao NEH original. A população inicial é composta por um indivíduo gerado pela heurística NEH, (B_i% - 1) indivíduos gerados pela heurística NEH

modificada e os (100 – B_i)% indivíduos restantes são gerados aleatoriamente. O parâmetro B_i indica o percentual de indivíduos gerados eficientemente.

Duas condições foram desenvolvidas para controlar como os indivíduos gerados substituem os indivíduos da população atual. A primeira é que um indivíduo gerado só substitui o indivíduo da população atual com o pior *makespan* se o seu *makespan* for menor e a segunda condição é que a seqüência do novo indivíduo seja única em relação a população atual. Isto ajuda a manter a diversidade na população.

Ruiz *et al.* (2006) desenvolveram quatro novos operadores de *crossover* para o PFSP que são baseados na idéia de identificar e manter os bons blocos construídos. O primeiro operador foi chamado de *Similar Job Order Crossover* (SJOX) que funciona da seguinte maneira. Os dois pais são examinados posição por posição. Quando as tarefas são idênticas na mesma posição elas são copiadas para os dois descendentes (Figura 2.5), depois é escolhido um ponto de corte aleatoriamente e cada um dos descendentes herda todas as tarefas a esquerda do ponto de corte de um dos pais (Figura 2.6) e finalmente as tarefas que faltam em um dos descendentes são copiados em ordem relativa do outro pai (Figura 2.7). O segundo operador de *crossover* foi resultado da constatação de que algumas vezes muitas tarefas iguais isoladas apareciam, por isso, desenvolveram outro operador de *crossover* chamado *Similar Block Order Crossover* (SBOX). A única diferença do SBOX em relação ao SJOX é que no primeiro passo só são copiados blocos idênticos de ao menos duas tarefas. O terceiro operador de *crossover* é chamado de *Similar Job 2-Point Order Crossover* (SJ2OX) que é similar ao operador SJOX com a diferença que são dois pontos de corte, ao invés de um. As tarefas entre dois pontos de corte são copiadas de um dos pais, enquanto os trabalhos dos dois extremos são preenchidos em ordem relativa pelas tarefas dos extremos do outro pai. O quarto operador de *crossover* é chamado de *Similar Block 2-Point Order Crossover* (SB2OX) que é similar ao operador SBOX só que utiliza dois pontos de corte.

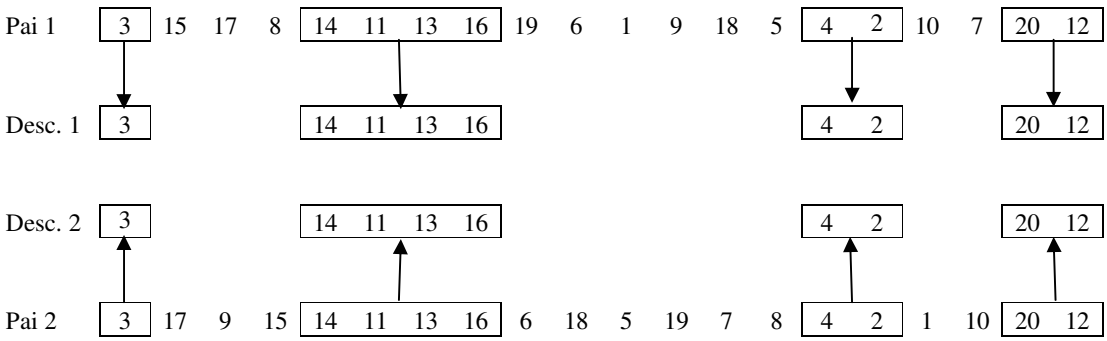


Figura 2.5 – Primeiro passo do *crossover* SJOX. Fonte: Ruiz *et al.* (2006).

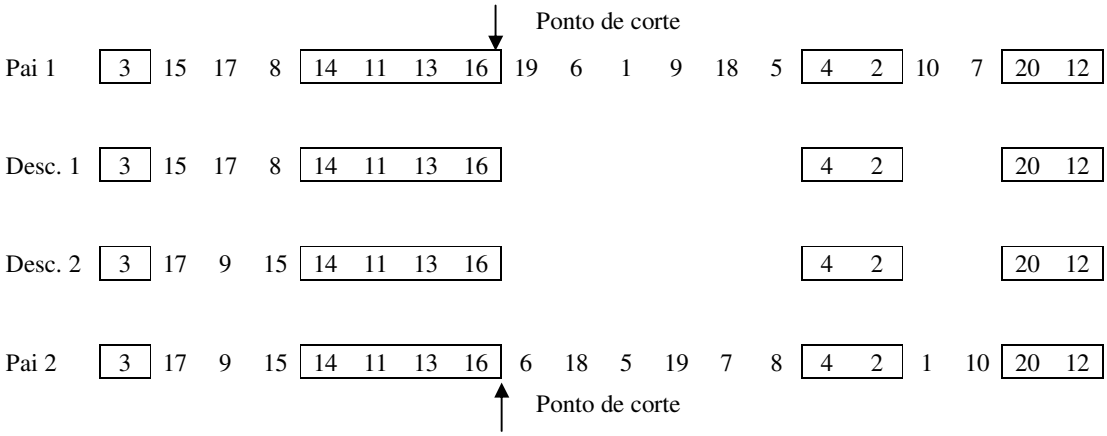


Figura 2.6 – Segundo passo do *crossover* SJOX. Fonte: Ruiz *et al.* (2006).

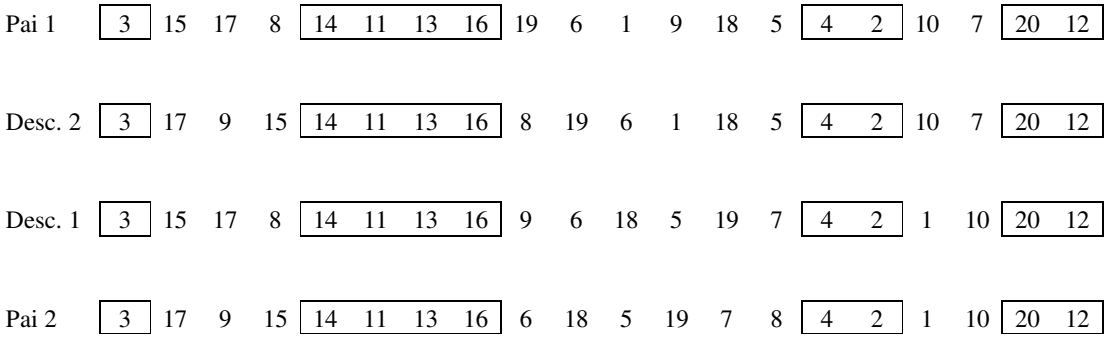


Figura 2.7 – Terceiro passo do *crossover* SJOX. Fonte: Ruiz *et al.* (2006).

O tipo de mutação implementada foi a *shift* que consiste em escolher uma tarefa aleatoriamente para ser colocado numa posição escolhida também aleatoriamente.

O procedimento *restart scheme* foi desenvolvido com o objetivo de evitar a convergência prematura do AG. Este procedimento é executado toda vez que um número de gerações sucessivas G_r são executadas e não é gerado um indivíduo melhor. Este procedimento é descrito a seguir.

1 - Colocar a população em ordem crescente em relação ao *makespan*;
2 - Manter os 20% dos melhores indivíduos;
3 - Gerar 40% de novos indivíduos a partir da mutação *shift* dos 20% melhores indivíduos;
4 - Gerar 20% dos indivíduos a partir da modificação da heurística NEH; e
5 - Gerar os 20% restantes dos indivíduos de forma aleatória.

A hibridização consiste da aplicação de uma busca local baseada na técnica *insertion neighborhood* que realiza todas as possíveis inserções e armazena a melhor seqüência. Se o resultado for melhor do que a seqüência atual a busca é repetida, senão a busca é encerrada. A busca local é realizada a cada geração em cada indivíduo com a probabilidade P_{enh}.

O primeiro AG não tem a etapa de busca local, ou seja, $P_{enh} = 0$ o segundo AG tem a etapa de hibridização, ou seja, $P_{enh} > 0$.

O projeto de experimentos consiste da comparação de todas as possíveis combinações de operadores genéticos e valores dos parâmetros. A seguir são apresentadas as combinações avaliadas.

- Tipo de seleção : *ranking* e torneio;
- Tipo de *crossover* : OP, OX, PMX, SB2OX, SBOX, SJ2OX, SJOX e TP;
- Probabilidade de *crossover* (P_c) : 0,0 – 0,1 – 0,2 – 0,3 e 0,4;
- Probabilidade de mutação (P_m) : 0,0 – 0,05 – 0,01 e 0,015;
- Tamanho da população (P_{size}) : 20, 30, 40, e 50;
- Restart (G_R) : 25, 50 e 75; e
- Probabilidade de melhoria local (P_{enh}) : 0,025 – 0,05 – 0,075 e 0,1.

O total de combinações avaliadas foram 2 x 8 x 5 x 4 x 4 x 3 x 4 = 15.360. Criou-se 68 instâncias combinando os valores de n e m, n = {20, 50, 80, ..., 440, 470, 500} e m = {5, 10, 15, 20}, seguindo a metodologia de Taillard (1993).

Os operadores utilizados e os valores finais dos parâmetros dos AG de Ruiz *et al.* (2006) foram os seguintes: tipo de seleção: torneio; tipo de *crossover*: SBOX; P_c : 40%; P_m : 1%; P_{size} : 20; G_r : 25; P_{enh} = 5% e B_i = 25%.

O resultado interessante foi que em comparação com os limites inferiores dos problemas gerados, a melhor combinação de operadores e valores dos parâmetros obteve um desvio de 3,22%, enquanto a pior combinação obteve 3,85%. Como a diferença não foi tão significativa Ruiz *et al.* (2006) afirmaram que isto se deveu a robustez dos AG desenvolvidos.

CAPÍTULO 3 – O PROBLEMA DE SEQÜENCIAMENTO PERMUTACIONAL CONTÍNUO *FLOWSHOP*

Este capítulo é composto de três seções que tratam do CPFSP. A primeira seção apresenta a definição do CPFSP. A segunda seção mostra o CPFSP modelado como um POCP. Finalmente, a terceira seção apresenta a descrição e análise de seis artigos recentes referente a métodos de resolução do CPFSP.

3.1. Definição do CPFSP

O CPFSP é um problema da classe FSP originado por pelo menos uma alteração em uma das suposições *J8* ou *P6*, dadas na Seção 2.1. A suposição *J8* permite que os trabalhos esperem entre máquinas consecutivas pelo processamento e a suposição *P6* garante que existe área suficiente para armazenar os trabalhos em espera. A definição do CPFSP é semelhante ao do PFSP com o acréscimo da restrição de que os trabalhos não podem esperar entre máquinas consecutivas. Dado um conjunto de n tarefas para serem processadas num conjunto de m máquinas, onde todas as tarefas usam a mesma ordem de processamento nas máquinas e depois de iniciada uma tarefa, ela não deve esperar por processamento entre duas máquinas consecutivas, i.e., as tarefas devem ser processadas continuamente, o tempo de processamento da tarefa i na máquina j é dado por p_{ij}, i = 1, 2, 3,..., m e j = 1, 2, 3,..., n. O CPFSP consiste em determinar uma seqüência específica das n tarefas que otimize um critério de desempenho estabelecido. Esta definição só é válida na prática se as demais suposições apresentadas na Seção 2.1 forem verdadeiras. A Figura 3.1 apresenta o gráfico de Gantt de um CPFSP. Nota-se no gráfico de Gantt que não existe folga entre o processamento de cada uma das tarefas.

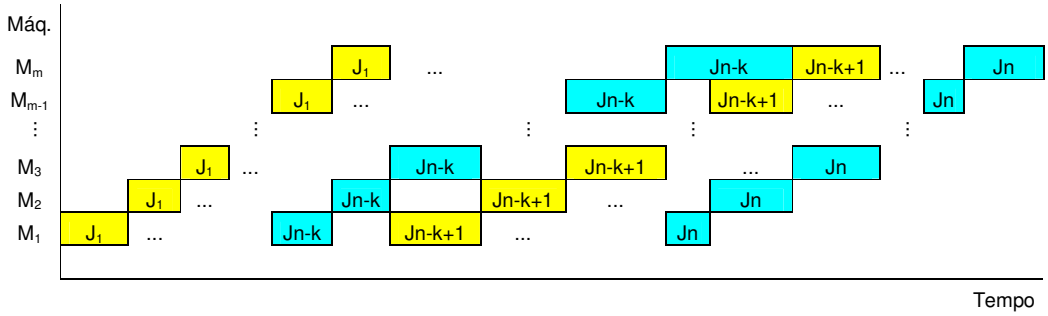


Figura 3.1 – Gráfico de Gantt de um CPFSP com n trabalhos e m máquinas.

3.2. O CPFSP como um POCP

O modelo matemático para o CPFSP não foi apresentado neste capítulo devido sua semelhança com o modelo matemático descrito para o PFSP, apresentado na seção 2.2, com a alteração apenas das restrições do grupo 2 que devem ser de igualdade.

Também é pequena a diferença do modelo POCP para o CPFSP em relação ao PFSP. Esta diferença está somente no cálculo da função *g*. Fink e Voβ (2003) apresentam uma fórmula para calcular o valor da função *g*. A seguir são apresentadas as duas fórmulas para calcular o valor de *g* de uma permutação *s* com critério de desempenho o tempo total de fluxo ou *makespan*.

$$\text{Tempo total de fluxo (TTF)}: \quad g_{CPFSP(TTF)} = \sum_{i=2}^{n} (n+1-i) d_{s(i-1),\,s(i)} + \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij} \qquad 3.1$$

$$\textit{Makespan}: \quad g_{CPFSP(\textit{makespan})} = \sum_{i=2}^{n} (n+1-i) d_{s(i-1),\,s(i)} + \sum_{j=1}^{m} p_{s(n)\,j} \qquad 3.2$$

A incógnita d_{ik} consiste no tempo de espera na primeira máquina entre o começo da tarefa *i* e o começo da tarefa *k*, quando *k* é processado posteriormente a *i*, em que $1 \leq i \leq n$ e $1 \leq k \leq n$, com $i \neq k$. A Equação 3.3 mostra como se calcula o valor de d_{ik}.

$$d_{ik} = \max_{1 \leq j \leq m} \{ \sum_{h=1}^{j} p_{ih} - \sum_{h=2}^{j} p_{k,\,h-1} \} \qquad 3.3$$

Segundo Röck (1984) os primeiros a analisarem a complexidade do CPFSP para m>2 foram Lenstra *et al.* (1977), seguidos por Papadimitriou e Kanellakis (1980) em que ambos provaram que o problema é NP-*hard* em sentido forte para m≥4. Existem casos que podem ser tratados polinomialmente, desde que os tempos de processamento satisfaçam uma estrutura especial dada por Panwalkar e Woollam (1980). Por fim, Röck (1984) provou para m≥3 que o CPFSP com o critério de minimização sendo o *makespan* é NP-*hard* em sentido forte.

3.3. Métodos de Resolução para o CPFSP

Foram avaliados seis artigos com propostas de métodos para a resolução do CPFSP. Os dois primeiros artigos usaram o critério de desempenho como sendo o tempo total de fluxo e os outros quatro artigos restantes usaram o *makespan*. Um dos artigos com o critério de desempenho sendo o tempo total de fluxo usou as instâncias de Taillard (1993) para testar os métodos propostos. Dois dos artigos com o critério de desempenho sendo o *makespan* usaram as instâncias de Reeves (1995) e Heller (1960) para testar os métodos propostos.

Os conceitos sobre o AG são apresentados no capítulo 4.

3.3.1. AG de Chen *et al.*

Chen *et al.* (1996) proporam dois AGs para o CPFSP com o tempo total de fluxo sendo o critério de desempenho. Um dos AGs foi implementado com população inicial eficiente e o outro com população inicial gerada aleatoriamente. A proposta principal do trabalho foi analisar a influência dos elementos do AG e os valores dos parâmetros de controle no desempenho do mesmo. Os AGs foram testados com problemas gerados aleatoriamente.

O trabalho foi dividido em duas partes, a primeira relacionada à escolha dos elementos do AG e a segunda relacionada à otimização dos parâmetros de controle. A seguir são apresentados os elementos que compõem o AG de Chen *et al.* (1996).

a) Representação do AG;
b) População inicial;
c) Tamanho da população;
d) Método de seleção;
e) Operadores genéticos; e
f) Critério de parada.

A representação genética adotada foi a permutacional. Por exemplo, para uma instância com $n = 8$ a estrutura pode ser representada por qualquer seqüência de oito trabalhos como

4 7 3 1 8 6 2 5. Segundo Cleveland e Smith (1989) *apud* Chen *et al.* (1996) vários operadores genéticos eficientes têm sido desenvolvidos para esse tipo de representação.

A população inicial é gerada a partir de diferentes procedimentos, com dois objetivos, melhorar a aptidão média e a diversificação da população inicial. O primeiro objetivo está relacionado com a redução do tempo computacional e o segundo objetivo com uma busca mais eficiente. No primeiro AG metade da população inicial é gerada aleatoriamente e a outra metade usando algumas heurísticas conhecidas. Nas heurísticas utilizadas existem duas para o PFSP, os métodos CDS e de Danninbring e uma para o CPFSP, o método *Job Insertion Based* (JIB) de Rajendran e Chaudhuri (1990). O procedimento que gera a população inicial é o seguinte: o primeiro membro é gerado pelo método JIB, os m-1 membros seguintes são gerados pelo método CDS, onde m é o número de máquinas, o membro $m + 1$ é gerado pelo método de Danninbring; se o número de membros gerados é menor que a metade do tamanho da população, então um membro é selecionada aleatoriamente e duas tarefas são escolhidas aleatoriamente e trocadas suas posições, dando origem a um novo membro. Esse procedimento é repetido até o número de membros ser igual à metade do tamanho da população. Para analisar o efeito da população inicial no desempenho do AG, foi implementado um segundo AG com a população inicial gerada totalmente de forma aleatória e comparado com o primeiro AG.

O procedimento apresentado a seguir é utilizado para determinar o valor da aptidão e da probabilidade de seleção de cada indivíduo da população.

1 – Calcular o tempo total de fluxo de cada indivíduo da população;
2 – Determinar o F_{max} que é o tempo total de fluxo máximo encontrado na população;
3 – Calcular o valor da aptidão de cada indivíduo, que é igual à diferença entre o F_{max} e o tempo total de fluxo do indivíduo; e
4 – Calcular a probabilidade de seleção de cada indivíduo, igual à divisão do valor da aptidão do indivíduo pela soma do valor das aptidões de todos os indivíduos da população.

A probabilidade de seleção de cada membro é usada como critério de seleção para os dois pais que participarão do processo de reprodução. O procedimento de seleção adotado foi o método de seleção por roleta.

Os operadores de *crossover* e mutação foram os operadores genéticos escolhidos para compor os AGs. O operador de *crossover* utilizado foi o PMX desenvolvido por Goldberg (1989) e descrito na seção 2.5.1. O operador de mutação adotado foi o *swap*, que troca dois trabalhos de posição aleatoriamente.

Existem dois fatores conflitantes a considerar no critério de parada: a intensidade da busca e o tempo computacional. Se a intensidade da busca é grande o tempo computacional é alto e a qualidade da solução é melhor, caso contrário, uma intensidade de busca menor exige menos tempo computacional, mas a qualidade da solução final pode ser pior. Por isso, foram usados dois critérios de parada para a busca: se o número de estruturas na população com o menor tempo de fluxo é maior que 60% da população ou o número de gerações é igual a 60.

A segunda parte da metodologia de Chen *et al.* (1996) foi otimizar os parâmetros de controle do AG. Foram adotados os mesmos parâmetros de Grefenstette (1986): tamanho da população (*N*), taxa de *crossover* (*C*), taxa de mutação (*M*), *gap* geracional (*G*), *scaling window* (*W*) e estratégia de seleção (*Se*). Assim, os parâmetros do AG são representados da seguinte forma AG (*N*, *C*, *M*, *G*, *W*, *Se*).

O *gap* geracional é o percentual da população que é trocada a cada geração. O *scaling window* é o número de gerações durante o qual o valor de f' é atualizado, onde f' é usado como uma base para calcular o valor de aptidão da cada estrutura. Para um problema de minimização o valor de f' é definido como o valor do objetivo máximo do indivíduo avaliado e o valor da aptidão de um indivíduo é definido como a diferença entre f' e o valor objetivo do indivíduo. O valor de *W* é definido para estar entre 0 e 7. Existem duas estratégias de seleção (*Se*): o indivíduo com a melhor aptidão é copiado para a próxima população (*Se* = *E*) ou todos os indivíduos da população são substituídos por novos indivíduos (*Se* = *P*).

DeJong (1980) *apud* Chen *et al*. (1996) depois de realizar experimentos para diversas combinações de valores dos parâmetros chegou aos seguintes valores para os parâmetros AG (50; 0.6; 0.001; 1.0; 7; *E*). Grefenstette (1986) *apud* Chen *et al*. (1996) desenvolveu

um AG para otimizar os parâmetros de outro AG e encontrou os seguintes valores AG (30; 0.95; 0.01; 1.0; 1; *E*).

Chen *et al.* (1996) testaram os valores ótimos dos parâmetros de DeJong (1980) e Grefenstette (1986). Eles observaram que ambos os conjuntos não se comportaram bem para o CPFSP. Por isso, Chen *et al.* (1996) modificaram o AG de Grefenstette (1986) e depois o utilizou para determinar os valores ótimos dos parâmetros de controle para o AG do CPFSP. Os novos valores encontrados foram $N = 95$, $C = 0.725$ e $M = 0.009$.

3.3.2. Metaheurísticas de Fink e Voβ

Fink e Voβ (2003) desenvolveram e analisaram vários métodos de resolução para o CPFSP com o critério de desempenho sendo o tempo total de fluxo. A implementação foi realizada com o *software* HotFrame (*Heuristic OpTimization FRAMEwork*) sem a preocupação de calibrar otimamente os parâmetros dos métodos. Para avaliar o desempenho dos métodos foram usadas as instâncias desenvolvidas por Taillard (1993).

Os métodos construtivos *nearest neighbor* (NN) e *cheapest insertion* (Chins) foram os primeiros a serem descritos por Fink e Voβ (2003). A heurística NN consiste em inserir em cada passo do método uma tarefa ainda não incluída com o mínimo tempo de espera para a última tarefa da seqüência em construção. A heurística Chins considera todas as possíveis inserções de todas as tarefas ainda não incluídas enquanto constrói uma seqüência completa, i.e, escolhendo uma tarefa inicial, em cada passo k, k= 2,..., n, a melhor combinação das n-k+1 tarefas em todas as k posições de inserção é determinada. A complexidade da heurística Chins é $O(n^3)$. A eficácia desses dois métodos construtivos depende da escolha da tarefa inicial.

Para melhorar a eficiência dos métodos construtivos foi utilizado o método *Pilot* desenvolvido por Duin e Voβ (1999) *apud* Fink e Voβ (2003) que consiste em considerar as conseqüências para o valor da função objetivo devido a escolha da inserção de uma nova tarefa, memorizando o melhor resultado e realizando todos os movimentos possíveis. Dessa forma consegue-se superar os usuais métodos míopes. O método *pilot* tem um parâmetro chamado extensão que define quantas posições da seqüência serão avaliadas

para todas as possíveis combinações, por exemplo, *pilot*-1 significa que para a primeira posição da seqüência as *n* tarefas são testadas com todas as possíveis inserções para as outras posições. O método *pilot* possui complexidade $O(n^6)$, por isso, para *n* grande o tempo computacional aumenta muito.

São descritos os movimentos *swap* e *shift* com complexidades $O(n^2)$ para gerar as vizinhanças. O movimento *swap* consiste na troca de pares de tarefa e o movimento *shift* consiste em inserir alguma tarefa numa nova posição. Esses dois movimentos são ilustrados na Figura 3.2.

Figura 3.2 – Movimentos *swap* e *shift*. Fink e Voβ (2003)

As estratégias gulosas *steepest descent* (SD) e *iterated steepest descent* (ISD) são implementadas com o objetivo de avaliar a qualidade dos movimentos *swap* e *shift*. O método SD consiste em selecionar e realizar em cada iteração o melhor movimento, a busca se encerra no ótimo local. Como a solução do ótimo local pode ser insatisfatória, o método ISD depois de encontrar um ótimo local utiliza algum esquema de perturbação para gerar uma nova solução inicial e recomeçar a busca.

Foram escolhidas as metaheurísticas *Tabu Search* (Glover e Laguna, 1997 *apud* Fink e Voβ, 2003) e *Simulated Annealing* (Kirkpatrick *et al.*, 1983 *apud* Fink e Voβ, 2003) porque são do tipo busca local e poderiam aproveitar os métodos construtivos e os tipos de movimentos.

A implementação dos métodos desenvolvidos também usou o *software* HotFrame. Os conceitos como problemas, soluções, vizinhanças e estratégias de diversificação são tratadas como objetos ou classes. O HotFrame gera estruturas para busca local e com a seleção de diferentes regras de vizinhança constrói os métodos SD e ISD. Do mesmo

modo, pode ser modificado para dar os métodos *simulated annealing* e *tabu search*. Também fornece componentes para representar os espaços de busca como estruturas de permutação. Fica a cargo do usuário, essencialmente, implementar a função objetivo. Para reduzir o tempo de execução, a avaliação dos movimentos *swap* e *shift* só é calculada a mudança realizada na seqüência. Por exemplo, usando um Pentium II (266 MHz) e um problema com $n = 200$, avaliou-se que o claculo de um movimento da forma direta se gasta 0.9 segundos, enquanto fazendo essa adaptação se gasta 0.05 segundos.

Os experimentos de Fink e Voβ (2003) mostram que o *simulated annealing* provê resultados com alta qualidade e baixo tempo de execução. Os métodos baseados no *tabu search* estático e estrito provêm resultados insatisfatórios. O *tabu search* reativo com solução inicial gerada pelo método pilot-10-Chins apresentou os melhores resultados entre todos os métodos avaliados.

3.3.3. Algoritmo Genético e *Simulated Annealing* de Aldowaisan e Allahverdi

Aldowaisan e Allahverdi (2003) desenvolveram quatro algoritmos de busca local para o CPFSP com o *makespan* como critério de desempenho. Dois desses algoritmos têm a solução básica inicial obtida pelo algoritmo *simulated annealing* (SA) de Chakravarthy e Rajendran (1999) e os outros dois têm a solução básica inicial obtida pelo AG de Chen *et al.* (1996). Para o processo de busca local foi desenvolvida uma nova heurística chamada de *insertion technique* (IT). Os algoritmos foram testados com problemas gerados aleatoriamente.

O método IT foi inspirado na heurística NEH criada por Nawaz *et al.* (1983). O método IT consiste em considerar dois trabalhos consecutivos como um bloco e o inserir em todas as posições ainda disponíveis na seqüência. O Quadro 3.1 apresenta a descrição do método IT.

Quadro 3.1 – Descrição do método IT.

Passo 1: Escolher uma seqüência *s*, onde os elementos são representados por *s* (*i*), *i* é a posição na seqüência e varia de 1 a *n*. *k* := 0.

Passo 2: *k* := *k* + 1. Selecionar *s* (*k*) e *s* (*k* + 1) para formar o bloco. Colocar o bloco nas posições de *k* a *n*. Para cada seqüência criada, trocar as posições de *s* (*k*) e *s* (*k* + 1) dentro do bloco e calcular o valor do *makespan*. Selecionar para a seqüência corrente a com menor *makespan*.

Passo 3: Repetir o passo 2 até *k* = *n* - 1.

Segundo Aldowaisan e Allahverdi (2003) até aquele momento a metaheurística *simulated annealing* ainda não tinha sido aplicada ao CPFSP. Eles optaram por adaptar o algoritmo SA desenvolvido por Chakravarthy e Rajendran (1999) para outro tipo de problema de *scheduling*. O AG usado por Aldowaisan e Allahverdi (2003) é o mesmo desenvolvido por Chen *et al.* (1996) para o CPFSP.

O Quadro 3.2 apresenta os pseudo-códigos das quatro buscas locais desenvolvidas por Aldowaisan e Allahverdi (2003). As buscas locais SA-1 e GEN-1 têm as soluções básicas iniciais obtidas pelos algoritmos SA e GEN, respectivamente. A busca local é implementada através da aplicação dos métodos NEH e IT alternadamente cinco vezes cada. Os experimentos realizados mostraram que dessa forma a qualidade da solução final era melhor que se os métodos fossem aplicados sucessivamente. Os experimentos também mostraram que a aplicação desses métodos mais de cinco vezes não proporcionava melhoria significativa na qualidade da solução final. As buscas locais SA-2 e GEN-2 têm as soluções básicas iniciais obtidas pelos algoritmos SA-1 e GEN-1, respectivamente. A busca local foi implementada através da aplicação do procedimento *pairwise* três vezes. O procedimento *pairwise* consiste em examinar cada possível troca de pares de uma tarefa numa posição com todas as outras tarefas. Os experimentos também mostraram que não havia melhoria significativa na qualidade da solução final quando o procedimento *pairwise* era aplicado mais de três vezes.

Quadro 3.2 – Os pseudo-códigos das buscas locais SA-1, SA-2, GEN-1 e GEN-2.

Algoritmo SA-1 (GEN-1)

Passo 1: Executar o algoritmo SA (GEN).

Passo 2: Fazer $s_0 = s^*$, onde s^* é a solução final obtida pelo algoritmo SA (GEN) e s_0 é a solução básica.

Passo 3: Aplicar sobre s_0 os métodos NEH e IT alternadamente cinco vezes cada.

Passo 4: A solução final é a melhor seqüência obtida no passo 3.

Algoritmo SA-2 (GEN-2)

Passo 1: Executar o algoritmo SA-1 (GEN-1).

Passo 2: Fazer $s_0 = s^*$, onde s^* é a solução final obtida pelo algoritmo SA-1 (GEN-1) e s_0 é a solução básica.

Passo 3: Aplicar sobre s_0 o procedimento *pairwise* três vezes.

Passo 4: A solução final é a melhor seqüência obtida no passo 3.

Aldowaisan e Allahverdi (2003) testaram os seus quatro algoritmos, o SA de Chakravarthy e Rajendran (1999), o AG de Chen *et al.* (1996) denominado de GEN, a melhor heurística desenvolvida por Gangadharan e Rajendran (1993) denominado de GAN-RAJ e a heurística desenvolvida por Rajendran (1994) denominada de RAJ. Para os testes foram usados 1.000 problemas gerados aleatoriamente com 20 combinações diferentes de 40 a 120 tarefas e 5 a 20 máquinas. Os algoritmos foram implementados em linguagem FORTRAN e os testes realizados num SUN SPARC Station 20. Os tempos de execução foram omitidos, mas segundo os autores o maior tempo de execução foi 10 segundos. O resumo dos resultados dos testes está apresentado na Tabela 3.1, na qual vê-se que os algoritmos propostos (SA-1, SA-2, GEN-1 e GEN-2) têm melhores desempenhos, já que as soluções iniciais são geradas a partir de heurísticas eficientes. O desempenho na média é semelhante para: SA-1 e GEN-1; e SA-2 e GEN-2.

Tabela 3.1 – Resumo dos resultados dos experimentos de Aldowaisan e Allahverdi (2003). Fonte: Aldowaisan e Allahverdi (2003).

Problemas	GAN-RAJ (%)	RAJ (%)	SA (%)	SA-1 (%)	SA-2 (%)	GEN (%)	GEN-1 (%)	GEN-2 (%)
40 x 5	4,97	6,15	3,29	0,96	0,42	6,18	0,75	0,31
40 x 10	4,91	5,24	2,66	0,63	0,44	6,32	0,60	0,26
40 x 15	5,41	4,79	2,56	0,46	0,18	5,78	0,88	0,58
40 x 20	5,09	4,42	2,87	0,57	0,30	6,04	0,57	0,34
60 x 5	5,26	6,52	2,98	1,09	0,35	6,22	1,12	0,26
60 x 10	5,64	4,50	2,15	0,79	0,53	5,70	0,56	0,20
60 x 15	5,85	4,66	2,32	0,50	0,18	6,01	0,65	0,41
60 x 20	5,99	5,06	2,44	0,54	0,23	5,76	0,74	0,39
80 x 5	5,48	7,04	3,26	1,15	0,29	6,20	1,17	0,18
80 x 10	6,32	4,78	2,28	0,58	0,25	5,63	0,50	0,21
80 x 15	6,99	4,81	2,18	0,42	0,18	5,82	0,50	0,19
80 x 20	7,04	4,83	2,14	0,61	0,28	5,33	0,57	0,23
100 x 5	5,83	6,99	3,37	1,20	0,22	5,69	1,41	0,48
100 x 10	6,51	4,73	2,10	0,78	0,40	5,17	0,59	0,27
100 x 15	7,37	4,83	1,91	0,66	0,33	5,39	0,46	0,19
100 x 20	8,00	4,31	1,84	0,48	0,16	5,22	0,43	0,20
120 x 5	6,61	7,77	3,52	1,40	0,30	5,61	1,39	0,20
120 x 10	6,76	4,56	2,09	0,65	0,31	4,67	0,52	0,16
120 x 15	7,54	4,34	1,50	0,44	0,18	4,84	0,41	0,21
120 x 20	7,64	4,31	1,75	0,57	0,27	4,93	0,48	0,19
Média	**6,26**	**5,23**	**2,46**	**0,72**	**0,29**	**5,63**	**0,72**	**0,27**

3.3.4. As Heurísticas de Aldowaisan e Allahverdi

Aldowaisan e Allahverdi (2004) desenvolveram também oito heurísticas para o CPFSP com o *makespan* como critério de desempenho. As heurísticas diferem em três aspectos: primeiro, a escolha entre os dois métodos de inserção; segundo, a escolha entre os dois critérios de parada; e finalmente, em usar ou não o procedimento de troca *pairwise*. As heurísticas propostas são comparadas às duas heurísticas de Rajendran e Chaudhuri (1990) e ao AG de Chen *et al.* (1996). As heurísticas foram testadas com problemas gerados aleatoriamente.

As heurísticas propostas por Aldowaisan e Allahverdi (2004) denotadas por PH*i* (*Proposed Heuristic*), onde *i*= 1, 2, 3 e 4, fazem uso da seqüência gerada pelo algoritmo ASI (Algoritmo de Seqüência Inicial), descrito no Quadro 3.3.

Quadro 3.3 – Descrição do algoritmo ASI.

Passo 1 : Para $k = 2$, $s_1 = \{1,..., n\}$ e $s_2 = \phi$.

Passo 2 : Escolher a tarefa i, tal que, $\sum_{j=1}^{m} p_{ij} \leq \sum_{j=1}^{m} p_{rj}, \forall\, r \in s_1$, onde p_{ij} é o tempo de processamento da tarefa i na máquina j. Remover a tarefa i de s_1 e colocar na primeira posição de s_2.

Passo 3 : Se $k = n$ ir para o passo 5, se não, calcular TTC_{1k} (tempo total de completação, considerando as tarefas da primeira posição até a k posição), para cada tarefa $i \in s_1$, depois de ser inserida na posição k. Remover a tarefa $i \in s_1$ que gerar o menor TTC_{1k} e inserir em s_2 na posição k. Atualizar $k = k+1$.

Passo 4 : Ir para o passo 3.

Passo 5 : Parar a iteração. A seqüência inicial é s_2.

As duas primeiras heurísticas propostas se diferenciam apenas pelo método de inserção utilizado no passo 3. A primeira heurística, PH1, usa o método de inserção NEH, desenvolvido por Nawaz *et al.* (1983). A segunda heurística, PH2, usa o método de inserção, RAZ, proposto por Rajendran e Ziegler (1997) *apud* Aldowaisan e Allahverdi (2004). O Quadro 3.4 apresenta a descrição das heurísticas PH1 e PH2.

Quadro 3.4 – Descrição das heurísticas PH1 e PH2.

Passo 1 : Gerar a seqüência inicial s_0 usando o algoritmo ASI. Determinar o valor da função objetivo T_0 da seqüência s_0, onde T é o valor do *makespan* da seqüência de tarefas.

Passo 2 : Atribuir $T_b = T_0$, $s_b = s_0$ e $r = 1$.

Passo 3 : Aplicar o método de inserção NEH (alternativamente, RAZ) para a seqüência s_{r-1} para obter s_r e calcular T_r.

Passo 4 : Se $T_r < T_b$, então, $T_b = T_0$ e $s_b = s_0$.

Passo 5 : Atualizar $r = r + 1$. Se $r > 10$ ir para o passo 6, caso contrário, ir para o passo 3.

Passo 6 : A seqüência do método PH1 (PH2) é s_b e o valor da função objetivo é T_b.

Os dois próximos métodos se diferenciam entre si pelo método de inserção e em relação aos dois primeiros métodos pelo procedimento de parada. A finalização nas duas primeiras heurísticas ocorre quando $r > 10$ e nos dois próximos métodos quando $r > 10$ ou $k = 2$. O Quadro 3.5 apresenta a descrição das heurísticas PH3 e PH4.

Quadro 3.5 – Descrição das heurísticas PH3 e PH4.

Passo 1 : Gerar a seqüência inicial s_0 usando o algoritmo ASI. Determinar o valor da função objetivo T_0 da seqüência s_0.

Passo 2 : Atribuir $T_b = T_0$, $s_b = s_0$, $r = 1$ e $k = 0$.

Passo 3 : Aplicar o método de inserção NEH (alternativamente, RAZ) para a seqüência s_{r-1} para obter s_r e calcular T_r.

Passo 4 : Se $T_r < T_b$, então, $T_b = T_0$, $s_b = s_0$ e $k = 0$.

Passo 5 : Se $T_r \geq T_b$, então, $k = k + 1$.

Passo 6 : Atualizar $r = r + 1$. Se $r > 10$ ou $k = 2$ ir para o passo 7, caso contrário, ir para o passo 3.

Passo 7 : A seqüência do método PH3 (PH4) é s_b e o valor da função objetivo é T_b.

A partir da incorporação do procedimento de troca *pairwise* às heurísticas anteriores, obtêm-se novas heurísticas denotadas por PH*i*(p), onde i= 1, 2, 3 e 4. O procedimento *pairwise* consiste em examinar cada possível troca de pares de uma tarefa numa posição com todas as outras tarefas.

Entre as oito heurísticas desenvolvidas por Aldowaisan e Allahverdi (2004) foram apresentados os resultados dos testes das heurísticas PH1, PH1(p), PH3, PH3(p), PH4 e PH4(p) e comparadas com as duas heurísticas de Rajendran e Chaudhuri (1990) denominadas de R-C1 e R-C2 e o AGChen desenvolvido por Chen *et al.* (1996). Os experimentos computacionais foram realizados com 750 problemas, gerados aleatoriamente com 5 combinações diferentes de 50 a 400 tarefas e 5 a 25 máquinas com 30 replicações em cada classe. Os algoritmos foram implementados em linguagem FORTRAN e os experimentos realizados num SUN SPARC Station 20. Os tempos de execução em segundos são apresentados na Tabela 3.2, enquanto um resumo dos resultados dos testes é apresentado na Tabela 3.3.

Através da Tabela 3.2 percebe-se que os tempos usados pelos métodos de Aldowaisan e Allahverdi (2004) foram muito maiores que os tempos usados pelos métodos comparados. Analisando a Tabela 3.3 vê-se que a heurística PH1(p) foi a que teve o melhor desempenho, devido à qualidade da solução inicial obtida pelo método PH1, que entre os métodos sem a etapa de melhoria foi o que obteve o melhor desempenho. A conclusão de Aldowaisan e Allahverdi (2004) foi que seus métodos são melhores que as heurísticas existentes para o CPFSP, entretanto foram usados tempos de execução muito grandes.

Tabela 3.2 – Tempos em segundos usados nos experimentos de Aldowaisan e Allahverdi (2004). Fonte: Aldowaisan e Allahverdi (2004)

Instâncias	R-C 1	R-C 2	AGChen	PH1	PH1(p)	PH3	PH3(p)	PH4	PH4(p)
50 X 5	0,002	0,001	0,076	0,125	0,169	0,245	0,287	0,779	0,824
50 X 10	0,003	0,006	0,177	0,336	0,416	0,514	0,591	1,409	1,468
50 X 15	0,004	0,004	0,283	0,351	0,480	0,562	0,660	1,761	1,831
50 X 20	0,006	0,004	0,200	0,249	0,319	0,397	0,449	1,163	1,234
50 X 25	0,003	0,005	0,193	0,289	0,367	0,486	0,584	1,494	1,553
100 X 5	0,006	0,005	0,116	0,450	0,661	0,690	0,879	1,979	2,156
100 X 10	0,008	0,008	0,149	0,926	1,301	1,463	1,803	4,325	4,704
100 X 15	0,009	0,013	0,280	1,300	1,774	2,289	2,723	6,693	7,167
100 X 20	0,010	0,013	0,325	1,893	2,366	2,681	3,192	7,704	8,229
100 X 25	0,011	0,016	0,404	2,851	3,518	4,092	4,759	12,049	12,690
200 X 5	0,030	0,032	0,279	3,800	5,427	5,756	7,360	16,843	18,444
200 X 10	0,020	0,039	0,539	6,906	10,033	12,291	15,303	36,413	39,444
200 X 15	0,050	0,066	0,713	12,576	16,553	19,163	23,099	55,752	59,713
200 X 20	0,050	0,054	0,887	11,519	15,749	21,810	25,999	64,252	68,424
200 X 25	0,057	0,061	1,007	19,572	25,182	32,587	38,107	95,903	101,502
300 X 5	0,082	0,082	0,603	12,640	17,995	18,790	24,153	56,301	61,636
300 X 10	0,086	0,128	0,989	31,625	41,927	42,396	52,745	126,214	136,519
300 X 15	0,163	0,130	1,278	42,280	56,824	68,449	82,772	202,780	217,018
300 X 20	0,134	0,155	1,461	56,244	71,401	77,191	92,499	228,679	243,898
300 X 25	0,135	0,154	1,864	67,604	91,099	126,488	147,363	344,600	365,291
400 X 5	0,138	0,178	0,930	12,440	20,896	21,421	29,677	63,181	71,620
400 X 10	0,209	0,174	1,518	47,918	66,250	74,861	93,013	225,292	243,747
400 X 15	0,219	0,199	2,024	86,691	113,899	122,128	149,299	366,211	393,371
400 X 20	0,211	0,251	2,186	99,298	129,642	143,791	174,137	428,389	458,337
400 X 25	0,237	0,234	2,907	126,053	167,323	199,553	240,660	685,406	727,642
Média	**0,075**	**0,080**	**0,855**	**25,837**	**34,463**	**40,004**	**48,485**	**121,423**	**129,938**

Tabela 3.3 – Resumo dos resultados dos experimentos de Aldowaisan e Allahverdi (2004). Fonte: Aldowaisan e Allahverdi (2004).

Instâncias	R-C 1	R-C 2	AGChen	PH1	PH1(p)	PH3	PH3(p)	PH4	PH4(p)
50 X 5	4,532	2,739	2,635	1,034	0,407	1,110	0,469	1,721	0,973
50 X 10	2,932	2,016	1,624	0,250	0,108	0,325	0,147	1,202	1,003
50 X 15	3,272	2,414	1,936	0,188	0,053	0,301	0,142	1,857	1,589
50 X 20	2,782	2,698	2,040	0,261	0,090	0,357	0,199	1,455	1,331
50 X 25	3,576	2,980	2,323	0,323	0,078	0,442	0,183	1,572	1,445
100 X 5	5,366	3,496	3,484	1,127	0,358	1,486	0,601	1,550	0,661
100 X 10	3,853	2,606	2,404	0,320	0,103	0,557	0,295	1,465	1,152
100 X 15	3,903	3,072	2,827	0,260	0,011	0,498	0,251	1,944	1,797
100 X 20	4,294	3,147	2,930	0,149	0,012	0,629	0,409	2,184	1,988
100 X 25	3,941	3,380	3,008	0,169	0,021	0,451	0,267	2,222	2,116
200 X 5	6,484	3,596	3,593	1,563	0,383	1,928	0,571	1,683	0,359
200 X 10	4,361	2,635	2,627	0,300	0,036	0,555	0,209	1,499	1,113
200 X 15	3,667	2,455	2,442	0,251	0,032	0,575	0,294	1,972	1,754
200 X 20	3,819	2,843	2,806	0,166	0,000	0,585	0,416	1,980	1,855
200 X 25	3,888	3,161	3,054	0,217	0,003	0,557	0,334	2,288	2,129
300 X 5	6,995	3,678	3,673	1,631	0,215	1,949	0,344	1,796	0,222
300 X 10	4,300	2,475	2,474	0,343	0,039	0,667	0,293	1,571	1,173
300 X 15	3,914	2,493	2,492	0,212	0,000	0,614	0,366	2,137	1,934
300 X 20	3,698	2,734	2,717	0,172	0,003	0,398	0,197	2,186	2,023
300 X 25	3,880	2,862	2,841	0,212	0,002	0,572	0,335	2,133	1,999
400 X 5	7,065	3,832	3,829	2,051	0,248	2,289	0,389	2,168	0,322
400 X 10	4,163	2,325	2,323	0,385	0,027	0,675	0,253	1,247	0,786
400 X 15	3,635	2,273	2,266	0,209	0,005	0,515	0,268	1,934	1,717
400 X 20	3,788	2,585	2,554	0,191	0,003	0,558	0,321	2,158	2,004
400 X 25	3,635	2,814	2,764	0,225	0,006	0,418	0,257	2,196	2,037
Média	**4,230**	**2,852**	**2,707**	**0,488**	**0,090**	**0,760**	**0,312**	**1,845**	**1,419**

3.3.5. GASA de Shuster e Framinan

Wang e Zeng (2001) desenvolveram um método híbrido chamado de GASA para resolver o *Job Shop Scheduling Problem* (JSSP). O GASA é uma combinação das técnicas AG e *Simulated Annealing* (SA). Schuster e Framinan (2003) adaptaram o GASA para o CPFSP com o critério de desempenho sendo o *makespan*, e usaram as instâncias de Reeves (1995) e Heller (1960) para os testes.

Wang e Zheng (2001) criaram um novo operador de *crossover* para ser usado no GASA. Neste operador primeiramente um conjunto {1, 2, ..., n} é dividido em dois sub-conjuntos A_1 e A_2 aleatoriamente, sendo que cada sub-conjunto tem que possuir ao menos um elemento. Cada elemento contido num sub-conjunto é copiado para um descendente na mesma posição que ocupava no indivíduo pai. Depois são escolhidos aleatoriamente dois

indivíduos da população atual para serem os pais s_1 e s_2. Os descendentes s'_1 e s'_2 são criados da seguinte forma: s'_1 herda os elementos de s_1 pertencentes a A_1 e os elementos de s_2 pertencentes a A_2; s'_2 herda os elementos de s_1 pertencentes a A_2 e os elementos de s_2 pertencentes a A_1.

O GASA inicia com uma população inicial de tamanho P_{size} gerada aleatoriamente, uma temperatura inicial t_0 e o critério de parada é o número L de iterações sem melhoria. A cada iteração os procedimentos de *crossover*, mutação e SA são aplicados. Estes três procedimentos são descritos a seguir:

1 – *Crossover*: O melhor indivíduo da população e um indivíduo escolhido aleatoriamente são submetidos ao novo operador de *crossover*. Este procedimento é repetido $P_{size}/2$ vezes, gerando P_{size} novos indivíduos. Os novos indivíduos e a população corrente somam um total de $2*P_{size}$ indivíduos que são avaliados e os P_{size} indivíduos com melhores aptidões são submetidos ao processo de mutação.

2 – Mutação : A mutação consiste em escolher um intervalo entre $\{1, ..., n\}$ e inverter a ordem das tarefas no intervalo. Os P_{size} indivíduos gerados são avaliados juntamente com os P_{size} indivíduos originais, então os P_{size} indivíduos com as melhores aptidões são submetidos ao procedimento SA.

3 – Procedimento SA : O objetivo deste procedimento é realizar uma busca local em cada indivíduo da população. O SA é implementado da seguinte forma: são escolhidas duas posições aleatoriamente na seqüência do indivíduo e as respectivas tarefas são trocadas de posição, os valores dos *makespan* antes e depois da troca são armazenados, a nova solução é aceita com uma certa probabilidade dependendo da diferença entre os *makespan* antes e depois da modificação e a temperatura atual. Este procedimento é repetido $n * m$ vezes. A temperatura t decresce seguindo uma função de resfriamento exponencial $t_k = \lambda * t_{k-1}$, $\lambda \in \{0, 1\}$.

Os valores dos parâmetros do GASA foram:

- Tamanho da população (P_{size}) : 40;
- Fator de resfriamento (λ) : 0,9;
- Temperatura inicial (t_0) : - ($C_{worst} - C_{best}$) / ln(0,1), onde: C_{worst} é o pior *makespan* da população e C_{best} é o melhor *makespan* da população;
- Número de iterações sem melhoria (L) : 30;

Para comparar o desempenho do GASA, Schuster e Framinan (2003) o testaram nas instâncias de Reeves (1995) e Heller (1960) e compararam com os resultados obtidos pela heurística RAJ de Rajendran (1994). O GASA foi implementado em linguagem C++ e os experimentos foram realizados num computador Athlon 1.400 MHz. Os resultados dos testes e os tempos utilizados pelo GASA são apresentados na Tabela 3.4, onde se verifica que de modo geral o GASA obtém melhores resultados do que o método RAJ, só que para isso precisa usar uma grande quantidade de tempo de execução. Além disto para as instâncias com maior número de tarefas e máquinas o GASA fica abaixo do método RAJ.

Tabela 3.4 – Resultados dos experimentos com o GASA. Fonte: Schuster e Framinan (2003).

Instância	*n* x *m*	RAJ	GASA	t (s)	Desvio (%)
rec01	20x5	1590	1527	6	-3,96
rec03	20x5	1457	1392	6	-4,46
rec05	20x5	1637	1524	7	-6,90
rec07	20x10	2119	2046	12	-3,45
rec09	20x10	2141	2045	11	-4,48
rec11	20x10	1946	1881	10	-3,34
hel2	20x10	189	180	10	-4,76
rec13	20x15	2709	2556	17	-5,65
rec15	20x15	2691	2529	17	-6,02
rec17	20x15	2740	2590	16	-5,47
rec19	30x10	3157	2985	34	-5,45
rec21	30x10	3015	2948	35	-2,22
rec23	30x10	3030	2827	35	-6,70
rec25	30x15	3835	3732	55	-2,69
rec27	30x15	3655	3560	51	-2,60
rec29	30x15	3583	3440	54	-3,99
rec31	50x10	4631	4757	147	2,72
rec33	50x10	4770	4998	145	4,78
rec35	50x10	4718	4891	146	3,67
rec37	75x20	8979	9508	907	5,89
rec39	75x20	9158	9964	890	8,80
rec41	75x20	9344	9978	904	6,79
hel1	100x10	780	877	1088	12,44
Média				**200,13**	**-1,18**

3.3.6. Os Algoritmos de Grabowski e Pempera

Grabowski e Pempera (2005) propuseram cinco algoritmos de busca local, dois deles baseados na técnica *Descending Search* (DS) e três baseados na metaheurística *Tabu Search* (TS), para resolver o CPFSP com o *makespan* sendo o critério de desempenho. As características mais importantes desses algoritmos são: o emprego de multimovimento e lista tabu dinâmica. A solução inicial de todos os algoritmos é obtida pelo método NEH de Nawaz *et al.* (1983). As instâncias de Reeves (1995) e Heller (1960) foram utilizadas nos experimentos computacionais para avaliar o desempenho dos métodos.

O tipo de movimento e a estrutura de vizinhança são componentes importantes dos algoritmos propostos. Um movimento é definido pelo par v = (x, y) que são duas posições da permutação *s*, com x, y $\in \{1, 2, ..., n\}$ e x $\neq$ y. Segundo Grabowski e Pempera (2005), baseados na literatura e em experimentos realizados, o movimento *shift* é o que melhor se adapta ao CPFSP, por isso, é adotado pelos algoritmos. A vizinhança da permutação *s* consiste das permutações s_v obtidas pela execução de todos os movimentos de um dado conjunto de movimentos Z e denotada por $N(Z, s) = \{ s_v \mid v \in Z\}$. Os algoritmos propostos geram vizinhanças através de movimentos $Z = \{ (x,y) \mid x, y \in \{1, 2, ..., n\}, y \notin \{x, x-1\}\}$ de cardinalidade $(n - 1)^2$, onde a condição $y \notin \{x, x-1\}$ evita a redundância de movimentos.

Para acelerar a convergência às boas soluções os algoritmos utilizam um procedimento chamado multimovimento, que tem o propósito de guiar a busca para regiões mais promissoras onde boas soluções podem ser encontradas. O multimovimento consiste de um conjunto de vários movimentos individuais que são executados simultaneamente numa única iteração do algoritmo. A execução do multimovimento gera permutações que diferem significativamente daquelas obtidas pela execução de um único movimento e conduz o processo de busca para regiões até o momento não visitadas do espaço de soluções. Segundo Grabowski e Pempera (2005) a aplicação de multimovimento em algoritmos de busca local é uma forma de adotar as estratégias de intensificação e diversificação no processo de busca. O multimovimento é composto de um conjunto de movimentos individuais proveitosos e independentes.

O subconjunto PZ = { $v \in Z \mid C_{max}(s_v) < C_{max}(s)$} é chamado de conjunto de movimentos proveitosos e contêm todos os movimentos do conjunto Z que geram permutações s_v com menores *makespan* que *s*. Dois movimentos $v_1 = (x_1, y_1) \in PZ$ e $v_2 = (x_2, y_2) \in PZ$ são chamados independentes em relação a permutação *s* se cada uma das posições x_1 e y_1 estão separadas de cada uma das posições x_2 e y_2 por pelo menos uma tarefa. Mais precisamente os movimentos v_1 e v_2 são independentes se alguma das condições 3.4 a 3.6 é satisfeita.

$$\begin{cases} \max(x_1, y_1) + 1 < \min(x_2, y_2) \\ \text{ou} \\ \max(x_2, y_2) + 1 < \min(x_1, y_1) \end{cases} \quad 3.4$$

$$\begin{cases} \min(x_1, y_1) + 1 < \min(x_2, y_2) \text{ e } \max(x_2, y_2) + 1 > \max(x_1, y_1) \\ \text{ou} \\ \min(x_2, y_2) + 1 < \min(x_1, y_1) \text{ e } \max(x_1, y_1) + 1 > \max(x_2, y_2) \end{cases} \quad 3.5$$

$$\begin{cases} \min(x_1, y_1) + 1 < \min(x_2, y_2) \text{ e } \min(x_2, y_2) + 1 < \max(x_1, y_1) \text{ e} \\ \max(x_1, y_1) + 1 < \max(x_2, y_2) \\ \text{ou} \\ \min(x_2, y_2) + 1 < \min(x_1, y_1) \text{ e } \min(x_1, y_1) + 1 < \max(x_2, y_2) \text{ e} \\ \max(x_2, y_2) + 1 < \min(x_1, y_1) \end{cases} \quad 3.6$$

A Condição 3.4 indica que os movimentos v_1 e v_2 operam em série em relação a permutação *s* e separados por pelo menos uma tarefa. A Condição 3.5 indica que v_1 opera do lado de dentro de v_2, ou vice-versa, com as posições x_1 e y_1 separadas de x_2 e y_2, por pelo menos uma tarefa. Finalmente a Condição 3.6 indica que v_1 e v_2 são interseccionados, com cada uma das posições x_1 e y_1 e separadas de x_2 e y_2 por pelo menos uma tarefa.

Define-se IPZ como sendo o subconjunto de PZ que contêm todos os movimentos independentes de PZ, isto significa que para cada par v_1 e $v_2 \in$ IPZ, $v_1 \neq v_2$, é satisfeita alguma das condições (3.4) a (3.6). O multimovimento então, consiste em executar todos os movimentos de IPZ simultaneamente, gerando uma permutação $s_{v'}$, onde $V' \in$ IPZ. A permutação $s_{v'}$ não pertence a N(Z, s), a menos que $|V'| = 1$. A seguir é apresentado o procedimento para criar o multimovimento V'.

Passo 1: Para uma dada permutação s, criar o conjunto PZ e atribuir V' := ϕ.

Passo 2: Encontrar o melhor movimento v*, i.e., $C_{max}(s_{v*}) = \min_{v \in PZ} C_{max}(s_v)$ e atribuir PZ := PZ – {v*} e V' := V' U {v*}.

Passo 3: Encontrar o melhor movimento v*, i.e., $C_{max}(s_{v*}) = \min_{v \in PZ} C_{max}(s_v)$ e para cada movimento v ∈ V' verificar as condições (3.4) a (3.6) para os movimentos v* e v. Se há um movimento v ∈ V' tal que para v* e v alguma condição não é satisfeita, então PZ := PZ – {v*}, caso contrário, PZ := PZ – {v*} e V' := V' U {v*}.

Passo 4: Repetir o passo 3 até PZ := ϕ.

Por intuição, $s_{v'}$ deveria ser significativamente melhor que s_v gerado pelo melhor movimento individual v ∈ V', desde que a melhoria total de $C_{max}(s_{v'})$ seja obtida pela adição de todos os melhoramentos produzidos pelos movimentos individuais de V'. Segundo Grabowski e Pempera (2005) essa é uma propriedade específica, conseqüência da restrição não espera da qual se utilizam os movimentos proveitosos e independentes e que não tem sido aplicado por outros métodos.

O primeiro algoritmo proposto foi um DS que consiste em pesquisar a vizinhança N até encontrar um movimento v* ∈ Z que gere uma permutação s_{v*} ∈ N com menor *makespan* que *s*, uma solução inicial. Dessa forma a permutação s_{v*} se torna a nova solução, i.e., $s := s_{v*}$ e o algoritmo é repetido até que nenhuma permutação melhor seja encontrada.

O segundo algoritmo proposto é uma combinação de DS e multimovimento (DS+M). O DS+M começa de uma solução inicial *s* e uma vizinhança N(Z, *s*). Para a vizinhança N o conjunto de multimovimentos V' é criado de acordo com o procedimento descrito anteriormente. O multimovimento V' é realizado e a permutação resultante $s_{v'}$ se torna a nova solução, i.e., $s := s_{v'}$, o algoritmo é repetido até que V' = ϕ.

Segundo Grabowski e Pempera (2005) a metaheurística *Tabu Search* (TS) não tinha sido aplicada até então no CPFSP. Assim foram desenvolvidos três algoritmos baseados em TS que têm como principal característica à utilização de lista tabu dinâmica com o propósito de evitar que o processo de busca fique preso a um ótimo local.

O comprimento da lista T é alterado quando o número de iterações (*iter*) do TS atinge um valor específico chamado de *pick*. Esse tipo de lista foi empregado primeiramente no *very fast* TS, proposto por Grabowski e Wodecki (2004).

Em relação a permutação *s*, um movimento (x, y) $\in$ Z é proibido: se A(*s*(x)) $\cap$ {*s*(x+1), *s*(x+2), ..., *s*(y)} $\neq \phi$, se x < y; ou B(*s*(x)) $\cap$ {*s*(y), *s*(y+1), ..., *s*(x-1)} $\neq \phi$ se x > y. Onde: A(*j*) = { $i \in$ J | (*j*, *i*) $\in$ T} e B(*j*) = { $i \in$ J | (*i*, *j*) $\in$ T}. O conjunto A(*j*) (ou B(*j*)) indica quais tarefas são processadas depois (ou antes) da tarefa *j* em relação ao conteúdo atual da lista tabu T.

O TS começa de uma solução básica inicial *s* a qual é aplicada à vizinhança N(Z, *s*). Primeiramente, o melhor movimento v* $\in$ Z que gera a permutação $s_{v*} \in$ N(Z, *s*) com o menor *makespan* é escolhido. Se $C_{max}(s_{v*}) < C^*$, então o movimento v* é selecionado para o processo de busca. Caso contrário ($C_{max}(s_{v*}) \geq C^*$), então é criado o conjunto *UZ* de movimentos não proibidos (*UF*) que não tem o status tabu e definido como *UZ* = { v $\in$ Z | movimento v é *UF*}. No próximo passo, $s_{v*} \in$ N(*UZ*, s) com o menor *makespan* é escolhido para o processo de busca. Se o movimento v* é selecionado, então o par de tarefas correspondentes ao movimento v* é adicionado à lista tabu T e a permutação resultante s_{v*} é criada. No passo seguinte, a permutação se torna a nova solução, i.e., *s* := s_{v*} e o algoritmo começa a próxima iteração. Se todos os movimentos de Z são proibidos, um caso muito raro, i.e., *UZ* = ϕ, então o elemento mais velho da lista tabu T é retirado dela e a busca é repetida até que um movimento *UF* possa ser encontrado.

O quarto algoritmo proposto é uma combinação de TS e multimovimento (TS+M). O algoritmo TS+M é similar ao TS exceto que em cada iteração um multimovimento V', que contém vários movimentos simples, é realizado, ao contrário de um movimento simples v*. Se numa iteração do TS+M, o multimovimento V' contêm não mais que um movimento, i.e., |V'| $\leq$ 1, então o TS+M se transforma em TS.

O quinto e último algoritmo é uma combinação de TS e multimovimento (TS+MP) que é realizado somente em situações específicas. O multimovimento é realizado a cada vez que um número de iterações (*Piter*) onde não ocorre melhoria no *makespan* é atingido. Se *Piter*

é um número suficientemente grande o multimovimento nunca será criado e TS+MP se transforma em TS. O parâmetro *Piter* foi calibrado experimentalmente.

Para comparar o desempenho dos seus algoritmos, Grabowski e Pempera (2005) realizaram testes nas instâncias de Reeves (1995) e Heller (1960) e compararam com os resultados obtidos pela heurística RAJ de Rajendran (1994) e o GASA de Shuster e Framinan (2003). Os algoritmos foram implementados em linguagem C++ e os experimentos foram realizados em um Pentium 1000. O resumo dos resultados dos experimentos computacionais é apresentado na Tabela 3.5. Analisando esta tabela verifica-se que o melhor desempenho foi obtido pelo algoritmo TS-M.

Tabela 3.5 – Resumo dos resultados dos experimentos de Grabowski e Pempera (2005). Fonte: Grabowski e Pempera (2005).

Métodos	Desvio (%)	Tempo (s)
RAJ	0,00	-
GASA	-1,18	200,13
DS	-4,51	0,02
DS-M	-4,53	0,00
TS	-6,50	0,86
TS-M	-6,59	0,87
TS-MP	-6,56	0,87

CAPÍTULO 4 – ALGORITMO GENÉTICO

Este capítulo é composto de três seções. A primeira seção faz uma introdução sobre AG. A segunda seção apresenta os principais componentes de um AG. E finalmente, a terceira seção descreve o rAG.

4.1. Introdução

Segundo Dréo *et al.* (2006) o AG tem características como: diversificação que é explorar regiões do espaço de busca raramente visitadas; intensificação que é verificar quase completamente regiões do espaço de busca promissoras; e a memorização da melhor solução encontrada até o momento. Uma desvantagem está no processo de calibração dos seus parâmetros. Resultados teóricos disponíveis não são suficientes para ajudar no ajuste da calibração.

O AG foi criado por Jonh Holland durante as décadas de 1960 e 1970 (Holland, 1975), para simular computacionalmente o fenômeno da seleção natural. Foi um aluno de Holland, Goldberg, o primeiro a aplicar o AG num problema de otimização, na área de projeto de gasodutos (Haupt e Haupt, 2004). Depois desta aplicação o AG passou a ser considerado uma técnica de busca baseada nos princípios da genética e seleção natural. O AG é formado por uma população de indivíduos que representam as soluções do problema. Os indivíduos são avaliados por uma função que atribui um valor chamado aptidão a cada indivíduo da população segundo sua qualidade em relação à função objetivo do problema. Os indivíduos são escolhidos por um procedimento inspirado na seleção natural para passarem por operações genéticas que resultam em descendentes que comporão a nova população. A Figura 4.1 mostra o fluxograma de um AG, segundo Reeves e Rowe (2002). Os estudos mostram que a nova população tem a tendência de ter indivíduos com aptidões melhores do que a população anterior (Mitchell, 1998; Haupt e Haupt, 2004). Este processo de gerar novas populações é chamado de geração. O melhor indivíduo da última população é a solução a ser apresentada para o problema.

É importante salientar que o AG trabalha com uma população de soluções. Pode-se considerar isso como várias buscas locais sendo feitas ao mesmo tempo, que é chamada de paralelismo implícito (Holland, 1975). A vantagem deste paralelismo é que o processo de busca melhora a capacidade de sair de mínimos locais, devido a uma pesquisa mais abrangente do espaço de busca. Porém, trabalhar com várias soluções ao mesmo tempo traz a desvantagem de precisar de mais tempo computacional para avaliar as funções que calculam a aptidão das soluções, que às vezes tornam o AG mais lento que os métodos de busca em vizinhança que só trabalham com uma solução de cada vez.

```
Escolha de uma população inicial
enquanto o critério de parada não é satisfeito faça
   repita
     se a condição do crossover é satisfeira então
       início
         seleciona os cromossomos pais;
         escolhe os parâmetros do crossover;
         executa o crossover;
       fim
     se a condição da mutação é satisfeita então
       início
         seleciona o(s) cromossomo(s) para a mutação;
         escolhe os parâmetros da mutação;
         executa a mutação;
       fim
      avalia a aptidão dos descendentes;
    até a quantidade de descendentes necessária;
    atualiza nova população;
fim_enquanto
```

Figura 4.1 – Pseudocódigo de um AG básico. Fonte: Reeves e Rowe (2002).

4.2. Os Elementos de um AG

Através da revisão bibliográfica foram escolhidos oito componentes como sendo os mais importantes num projeto de AG. Os oito componentes são elicitados a seguir e descritos no decorrer desta seção.

a) Escolha da representação para o AG;
b) Definição da função de aptidão;
c) Definição da população inicial;
d) Escolha do método de seleção;
e) Escolha dos operadores genéticos;
f) Escolha da estratégia geracional;
g) Escolha do critério de parada; e
h) Escolha dos valores dos parâmetros.

4.2.1. Representação para o AG

A representação para o AG é a forma como as soluções potenciais são codificadas para ser possível a aplicação dos operadores genéticos. Na representação para o AG os conceitos de genótipo, fenótipo, cromossomo, alelo e gene são importantes (Rothlauf, 2006). O genótipo representa toda a informação armazenada no cromossomo. O fenótipo é a aparência de um indivíduo que é resultado da informação contida no genótipo. Um cromossomo é uma *string* de certo comprimento onde toda a informação genética de um indivíduo está armazenada, cada cromossomo é constituído de muitos alelos. Alelo é a menor unidade de informação num cromossomo. Um gene é uma região do cromossomo constituído por um ou mais alelos que devem ser interpretados conjuntamente e que é responsável por uma propriedade específica do fenótipo. Estes conceitos estão ilustrados na Figura 4.2.

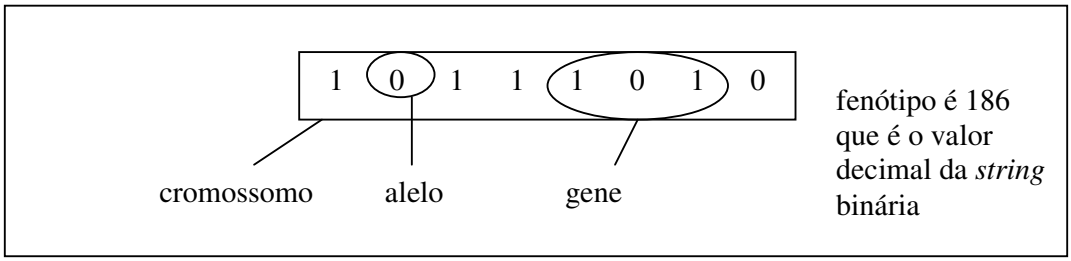


Figura 4.2 - Representação do fenótipo, cromossomo, alelo e gene.

A representação adotada num AG está diretamente relacionada ao tipo de problema. A primeira representação criada foi a binária onde os alelos podem assumir os valores 0 ou 1. Para muitos problemas de otimização combinatoria onde as variáveis do problema são binárias esta representação é ideal. Quando as variáveis do problema são contínuas a precisão depende do tamanho da *string* de alelos, quanto maior a *string*, maior o uso de recursos computacionais. Outro tipo de representação é a permutacional, onde os alelos podem assumir valores inteiros positivos e o cromossomo representa uma solução baseada na ordem dos alelos. Este tipo de representação é usado principalmente em problemas como o caixeiro viajante, seqüenciamento, entre outros.

4.2.2. Função de Aptidão

A aptidão corresponde ao grau de qualidade do fenótipo em relação ao seu habitat, i.e., significa o quanto o indivíduo está adaptado ao meio-ambiente. Para os problemas de otimização a aptidão significa a qualidade da solução em relação ao objetivo do problema. O valor da aptidão de cada indivíduo é muito importante, pois é usado para diferenciar os indivíduos na população e no processo de seleção. Se a aptidão não conseguir representar adequadamente a diferença entre os indivíduos, a eficácia do AG fica comprometida. A aptidão é calculada pela função de aptidão que pode ser uma função matemática, um experimento ou um jogo (Haupt e Haupt, 2004). Algumas vezes é usada a própria função objetivo como função de aptidão, esta estratégia pode ser ineficiente quando os valores da função objetivo dos indivíduos são muito próximos (Mitchell, 1998). Por isso, deve-se ter muito cuidado em se escolher a função de aptidão. Além disso, a função objetivo depende do problema abordado.

4.2.3. População Inicial

Segundo Reeves e Rowe (2002) as duas principais questões a considerar em relação a população inicial são o tamanho da população e o método usado para criar os primeiros indivíduos.

A principal idéia em relação à escolha do tamanho da população é a existência de um *trade-off* entre eficiência e eficácia. Parece lógico supor que para um certo comprimento de *string* que representa o indivíduo, exista um valor ótimo para o tamanho da população, não tão pequeno que não explore todo o espaço de busca e nem tão grande que comprometa o tempo de execução. Mas baseado no levantamento feito por Reeves e Rowe (2002) ainda não se determinou uma função que represente este suposto *trade-off*.

Em relação à escolha do método para gerar a população inicial as duas principais formas são a aleatória e a baseada em boas soluções conhecidas. A forma aleatória na prática é pseudo-aleatória, pois é gerada por *software* usando funções matemáticas. Neste tipo existe a possibilidade da população inicial não explorar todas as regiões do espaço de busca e, por isso, precisar de uma população maior. A população inicial gerada baseada em boas soluções conhecidas tem o objetivo de fazer o AG obter melhores soluções em um tempo de execução menor em comparação a inicialização aleatória. Neste método existe a possibilidade de convergência prematura para uma solução de baixa qualidade. Surry e Radcliffe (1996) *apud* Reeves e Rowe (2002) fizeram uma revisão das idéias sobre o processo de criação da população inicial e concluíram que havia uma tendência na inicialização eficiente de reduzir a qualidade da solução obtida em comparação com a inicialização aleatória.

4.2.4. Métodos de Seleção

Depois da criação da população inicial e atribuída uma aptidão a cada indivíduo, a próxima decisão é escolher o método de selecionar os indivíduos que darão origem à próxima geração. A principal característica de qualquer método de seleção é preferir os indivíduos com maior aptidão com o objetivo que a próxima população tenha uma aptidão maior (Mitchell, 1998). No processo de seleção existe uma relação de compensação, quanto

maior a pressão de seleção, ou seja, quanto maior a preferência por indivíduos de alta aptidão mais rapidamente a população converge para um ótimo local, do contrário quanto menor a pressão de seleção, mais lentamente a população evolui para boas soluções.

O primeiro método de seleção criado foi o método de seleção por roleta (Holland, 1975). Neste método cada indivíduo tem a probabilidade de seleção proporcional a sua aptidão em relação à população. A maneira mais comum de implementar este método de seleção é atribuir um número real a cada indivíduo igual a sua aptidão dividida pela aptidão total da população. Isto implica que cada indivíduo recebe um número maior que 0 e menor que 1, representando uma probabilidade e o somatório da probabilidade de todos os indivíduos sendo igual a 1. Depois que estes indivíduos são ordenados numa lista que pode ser representada graficamente como um disco, onde cada setor angular é proporcional a probabilidade do indivíduo ser selecionado. O processo de seleção consiste em gerar N números aleatórios entre 0 e 1, onde N é o tamanho da população. O intervalo que este número estiver contido na lista de probabilidades acumuladas indica que aquele indivíduo foi selecionado. Isto é como se uma roleta fosse girada e onde ela parasse indicasse o indivíduo selecionado. A Figura 4.3 ilustra este tipo de seleção, onde são mostrados cinco indivíduos com suas respectivas representações e aptidões. O gráfico de setor ao lado representa a probabilidade 1 e conseqüentemente cada setor circular representa a probabilidade do indivíduo correspondente ser selecionado.

Nº	Indivíduo	Aptidão
1	010010011	3,651
2	010110110	0,544
3	011010001	0,239
4	110110011	1,463
5	100111011	2,987

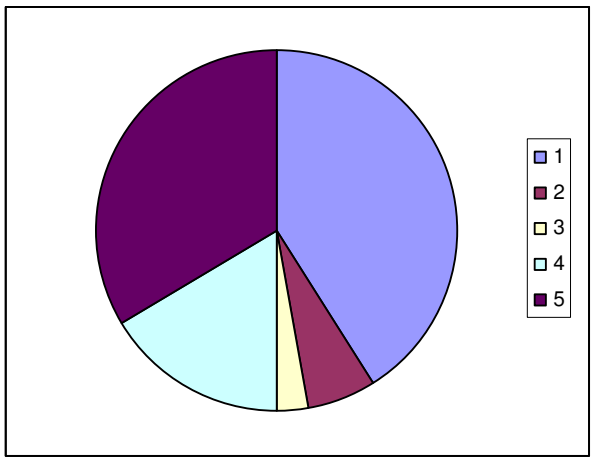


Figura 4.3 - Representação da seleção pelo método da roleta.

É comum ocorrer numa população uma pequena quantidade de indivíduos com alta aptidão, isto prejudica os métodos de seleção proporcionais porque a probabilidade destes indivíduos serem selecionadas é bem maior que do resto da população e, por isso, provoca a convergência prematura. Para evitar este problema outros métodos de seleção foram criados.

Outro método de seleção é o de seleção por torneio, mais resistente a convergência prematura. Na seleção por torneio um sub-conjunto *d* da população é escolhido aleatoriamente, um parâmetro predefinido *k* representa a probabilidade do melhor indivíduo do sub-conjunto ser escolhido. Neste método é gerado um número aleatório entre 0 e 1, se for menor que *k* o melhor indivíduo do sub-conjunto é escolhido, caso contrário outro indivíduo é escolhido (Mitchell, 1998). Este método tem a vantagem de usar pouco recurso computacional.

4.2.5. Operadores Genéticos

A função dos operadores genéticos é transformar a população atual numa nova população com aptidão melhor, ou seja, para problemas de otimização encontrar soluções melhores que as atuais (Mitchell, 1998). Um conceito importante neste processo é o bom bloco construído que é uma parte contínua do cromossomo que confere ao indivíduo uma alta aptidão. Acredita-se que durante a aplicação dos operadores genéticos os bons blocos construídos são formados e preservados, garantindo assim a qualidade das soluções. Os operadores genéticos dependem do tipo de representação adotada. Os principais operadores genéticos são o *crossover* e a mutação.

O operador mais utilizado é o *crossover*. Através do *crossover* são criados novos indivíduos misturando os alelos dos pais. O objetivo é que a mistura de bons blocos construídos dêem origem a indivíduos de aptidão melhor que os pais. Outro operador usado é o de mutação que consiste em trocar dois alelos de valor ou posição. A mutação tem o objetivo de manter a diversidade da população e evitar a convergência prematura para ótimos locais. A mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca não seja zero.

A implementação do operador *crossover* é feita normalmente com uma regra aleatória baseada numa distribuição uniforme. É definida uma probabilidade de ocorrência para o *crossover* chamada taxa de *crossover*. A forma mais comum de implementar a mutação é escolher um número de perturbações por *string* que é chamada de taxa de mutação.

O mais comum é que estas duas taxas sejam constantes durante todo o tempo de execução do AG. Mas Davis (1991) argumenta que se estas taxas variassem durante o processo de busca, melhores soluções seriam encontradas. Neste ponto de vista diferentes taxas seriam apropriadas em diferentes fases do processo de busca. No início seria usada uma taxa de *crossover* alta para fazer uma pesquisa maior no espaço de busca, enquanto que no fim uma taxa de mutação alta seria usada para diminuir a convergência da população. Outra sugestão é que as taxas dos operadores se adaptem instantaneamente em concordância com a evolução das soluções encontradas.

Ainda em Reeves e Rowe (2002), baseados no trabalho de Holland (1975), o operador de *crossover* sempre deveria ser usado. Mas existem na literatura duas estratégias para gerar a próxima população: *crossover*-e-mutação e *crossover*-ou-mutação. Na primeira estratégia o operador de *crossover* é aplicado com uma probabilidade normalmente menor que 1 e a mutação pode ser realizada se o seu critério for verdadeiro. Nesta estratégia existe a possibilidade dos filhos serem apenas clones dos pais devido a probabilidade de nem um dos operadores serem aplicados. Na segunda estratégia sempre um dos operadores é aplicado, ou *crossover* ou mutação, mas não ambos. Nesta estratégia não existe a possibilidade dos filhos serem clones dos pais.

4.2.6. Estratégia Geracional

A estratégia geracional é responsável por controlar a substituição de indivíduos de uma geração para a outra. A estratégia geracional proposta por Holland (1975) cria um conjunto do tamanho da população de indivíduos gerados a partir da população atual, usando os operadores de *crossover* e mutação. No final este conjunto substitui a população atual. Neste tipo de estratégia existe a possibilidade de que bons indivíduos desapareçam de uma geração para a outra. Por isso, surgiram outras estratégias como a elitista, a *population overlaps* e a *steady-state*. Na estratégia elitista o melhor indivíduo é preservado para a

próxima população, enquanto o restante da população é substituída por novos indivíduos. Na estratégia *population overlaps* uma fração da população G (*generation gap*) é substituída por novos indivíduos, enquanto a outra fração é preservada para a próxima população. Na estratégia *steady-state* só o melhor indivíduo gerado é copiado para a próxima população.

4.2.7. Critério de Parada

Nesta seção será feita a descrição da escolha de qual critério será usado para finalizar a execução do AG. Nos métodos de busca em vizinhança, que trabalham com somente uma solução, uma alternativa é encerrar a execução quando um ótimo local é obtido, mas no AG isso não é possível. As três estratégias mais comuns para encerrar a execução de um AG são:

i) O número de gerações;
ii) O tempo de execução; e
iii) A diversidade da população. Quando a semelhança entre os indivíduos começa a se repetir, então é definido o momento de parar, por exemplo, quando 90% dos indivíduos são semelhantes.

4.2.8. Parametrização do AG

A última decisão num projeto de AG é a definição dos valores dos seus parâmetros, como tamanho da população, taxa de *crossover* e taxa de mutação. Segundo Mitchell (1998) os parâmetros dos AG interagem entre si de forma não-linear. Sendo assim, não podem ser otimizados ao mesmo tempo. Muitos trabalhos têm sido realizados nesta área, entretanto nenhuma função matemática foi apresentada e que forneça os melhores valores para esses parâmetros (DeJong, 1980; Grefenstette, 1986 e Ruiz *et al.*, 2006). Como já mencionada anteriormente uma desvantagem do AG é a dificuldade do processo de calibração dos seus parâmetros (Dréo *et al.*, 2006). Diante disso, foi criado até um AG que utiliza poucos parâmetros por Lobo e Goldberg (2004), denominado de AG com menos parâmetros.

4.3. Descrição do rAG

Nesta seção é descrito o projeto do rAG. Para guiar a descrição foi usado o modelo visto na seção anterior.

4.3.1. Escolha da Representação

Devido aos dois problemas abordados neste trabalho, a representação mais adequada é a permutacional, onde os alelos são representados pelos números das tarefas e a ordem relativa das tarefas na permutação indica a ordem de processamento das mesmas nas máquinas.

4.3.2. Função de Aptidão

Para calcular a aptidão dos indivíduos da população foi adotado o valor da função objetivo de cada problema. Para o CPFSP foram adotadas duas aptidões porque foram testados problemas com duas funções objetivo diferentes. O rAG usou no CPFSP com o tempo total de fluxo como critério de desempenho a Equação 3.1, Seção 3.2. O rAG também usou com o *makespan* como critério de desempenho a Equação 3.2, Seção 3.2. E finalmente, o rAG utilizou no PFSP, o *makespan* como critério de desempenho, o procedimento g descrito na seção 2.3 para calcular a aptidão dos indivíduos da população. Usar o mesmo valor da função objetivo sem fazer nenhuma conversão, como outros AG fazem, para representar a aptidão dos indivíduos da população, foi uma das formas encontradas para diminuir o tempo computacional utilizado pelo rAG. Diante disso é possível diminuir a diferença, em tempo de execução, com relação aos métodos que trabalham sobre uma solução de cada vez. Mas esta estratégia não comprometeu a qualidade na diferenciação dos indivíduos que é o propósito da aptidão. Os métodos usados para calcular a aptidão dos indivíduos foi a única modificação na estrutura do rAG para permitir sua aplicação nos dois problemas deste trabalho. Isto mostra a generalização do uso do rAG na classe de problemas de sequenciamento permutacional.

4.3.3. População Inicial

Como uma das propostas do trabalho é a geração da população inicial completamente aleatória esta foi a forma escolhida. Além da aleatoriedade, evitar que indivíduos com a mesma seqüência de tarefas estejam presentes na população inicial também foi tratado. Esta prática melhora a diversidade, pois um indivíduo repetido agora gera um novo indivíduo para a população. Para eliminar os indivíduos repetidos da população inicial, foi implementado um procedimento que é executado depois da geração de todos os indivíduos da população. Este procedimento consiste em analisar todos os indivíduos da população e em encontrando um indivíduo repetido fazer ele passar pelo processo de mutação até que se torne um indivíduo único na população.

4.3.4. Método de Seleção

O tipo de seleção implementada no rAG foi a seleção por torneio porque é resistente a convergência prematura e tem custo computacional baixo. É escolhido um sub-conjunto com *d* (parâmetro descrito na seção 4.2.4) indivíduos e gerado um número aleatório entre 0 e 1, quando esse número for maior que *k* (parâmetro descrito na seção 4.2.4), o segundo melhor indivíduo do sub-conjunto *d* é escolhido. Nesta etapa, assim como também no processo de cálculo da aptidão dos indivíduos da população, foi levado em consideração a importância de reduzir o tempo computacional do rAG. A escolha deste método de seleção contribui para reduzir o consumo de tempo de execução do rAG e não comprometeu a eficácia na obtenção de boas soluções.

Nesta etapa, o segundo procedimento proposto para melhorar o desempenho do rAG foi implementado. O procedimento consiste em fazer que só um dos pais seja escolhido pelo método de seleção por torneio, o outro pai será o melhor indivíduo da população. Este procedimento foi denominado de *crossover* elistista e, a probabilidade de ocorrência é controlada pelo parâmetro P_{ce}. O objetivo deste procedimento é favorecer o processo de intensificação, já que o processo de *crossover* realizado com o melhor indivíduo da população tem grandes chances de gerar descendentes com alta aptidão.

4.3.5. Operadores genéticos

Nesta seção são descritos os operadores genéticos implementados no rAG. O operador *crossover* implementado foi o *Order Crossover* (OX) (Goldberg, 1989). O operador de mutação implementado foi o movimento *swap*. Nesta etapa foi implementado o terceiro procedimento, proposto para melhorar o desempenho do rAG. O novo operador genético é chamado de mutação populacional. Estes três operadores genéticos são descritos a seguir.

Na aplicação dos operadores genéticos foi adotada a estratégia *crossover*-ou-mutação, i.e., sempre um dos operadores é aplicado, ou *crossover* ou mutação, mas não ambos.

Crossover

O operador *crossover* OX foi criado baseado na idéia dos bons blocos construídos. Por isso, baseia-se nas posições relativa e absoluta das tarefas na seqüência. Este procedimento é descrito a seguir.

1 – São escolhidos dois pais através do método de seleção;
2 – É escolhido aleatoriamente o mesmo fragmento de cada um dos pais e copiado nos respectivos filhos (Figura 4.4). Esta etapa preserva as posições absoluta e relativa das tarefas na seqüência; e
3 – As posições não-preenchidas de cada filho são copiadas das tarefas do outro pai no sentido da esquerda para a direita (Figura 4.5). Este procedimento faz com que seja preservada a ordem relativa das tarefas na seqüência.

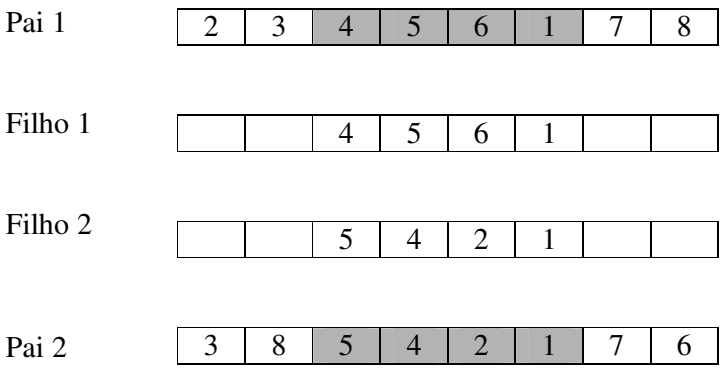


Figura 4.4 – Segunda etapa do *crossover* OX.

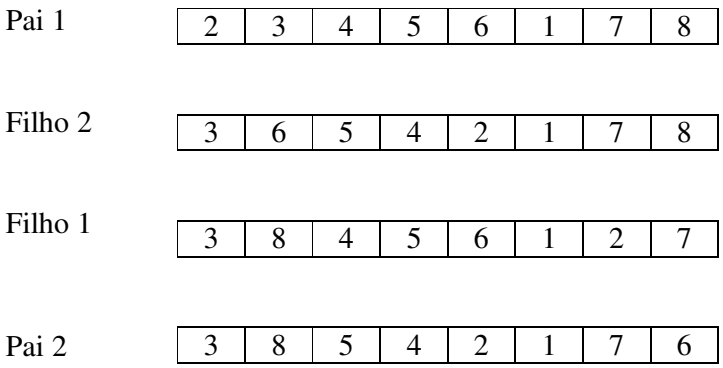


Figura 4.5 – Terceira etapa do *crossover* OX.

Mutação

O operador mutação *swap* consiste em realizar uma única alteração na estrutura do indivíduo. Este operador é implementado da seguinte forma: são escolhidos aleatoriamente duas posições na estrutura do indivíduo e o valor dos alelos dessas posições são trocados. Nesta etapa também existe a preocupação de evitar que surja na população um indivíduo repetido, por isso, quando o indivíduo gerado já existe na população o procedimento é repetido até a sua estrutura ser a única. A Figura 4.6 mostra um exemplo da aplicação deste operador para um problema com $n = 8$.

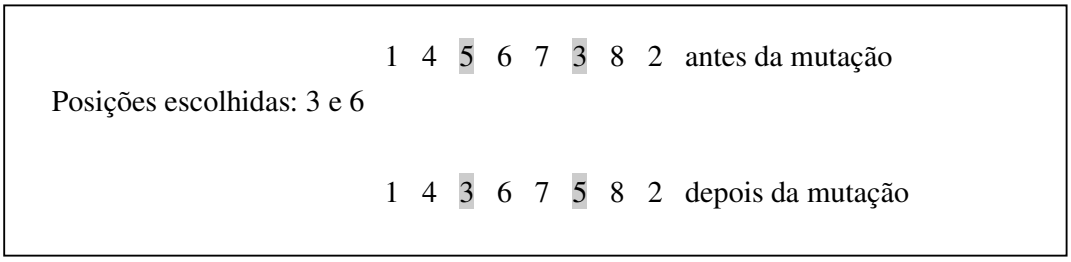


Figura 4.6 – Exemplo da aplicação do operador mutação (*swap*).

Mutação Populacional

A mutação populacional baseada no princípio da diversificação foi o terceiro procedimento proposto para melhorar o desempenho do rAG. A idéia deste operador surgiu na análise dos resultados do rAG, que demonstram que à medida que a qualidade da melhor solução aumenta mais gerações são necessárias para haver outra melhoria, isto prova, que o

processo de busca estava estagnando. A idéia foi executar uma perturbação em todos os indivíduos da população para reativar o processo de evolução, aproveitando a quantidade de gerações em que a melhor solução não se alterava. Para exemplificar a Tabela 4.1 mostra alguns valores dos resultados obtidos para a instância tai031 (Taillard, 1993), com 50 tarefas e 5 máquinas, pela versão do AG sem o procedimento criado. O tempo de execução do experimento foi de 3,75 segundos. Durante todo o processo de busca houve 174 melhorias. Os significados das colunas da Tabela 4.1 são: a coluna um representa o número da melhoria (NM); a coluna dois representa a aptidão do melhor indivíduo; a coluna três representa o número de gerações; a coluna quatro representa o número de gerações entre as melhorias (GEM); a coluna cinco representa o desvio $|(z - z^*)/z^*|$ (z : solução encontrada pelo rAG; z* : melhor solução conhecida para a instância) à solução final; e finalmente a coluna seis representa o percentual de gerações realizadas (P). Analisando os dados desta tabela se verifica que depois de 20,84% das gerações realizadas o valor de *GEM* pela primeira vez é maior que 100 e continua assim na maioria das vezes. Isso quer dizer que são centenas de gerações onde não ocorre melhoria da solução encontrada pelo rAG, por isso, desenvolveu-se um procedimento para reativar o processo de evolução da melhor solução encontrada, denominado mutação populacional.

Tabela 4.1 – Alguns valores obtidos para o problema tai031 por uma versão inicial do rAG.

NM	Aptidão	Geração	GEM	D (%)	P (%)
1	118932	0	0	54,77	0,00
5	108173	8	8	40,77	0,22
10	105199	19	11	36,90	0,53
15	99063	27	8	28,92	0,75
20	96612	42	15	25,73	1,17
25	94917	60	18	23,52	1,67
30	93790	85	25	22,06	2,37
35	91836	104	19	19,51	2,90
40	90859	126	22	18,24	3,51
45	89294	143	17	16,20	3,99
50	88469	191	48	15,13	5,33
55	87634	229	38	14,04	6,39
60	86805	248	19	12,97	6,92
65	86060	277	29	12,00	7,73
70	84948	324	47	10,55	9,04
75	84198	356	32	9,57	9,93
80	84115	441	85	9,46	12,30
85	83796	495	54	9,05	13,81
90	83184	554	59	8,25	15,45
95	82959	594	40	7,96	16,57
100	81861	654	60	6,53	18,24
110	81151	747	93	5,61	20,84
120	80044	1041	294	4,17	29,04
130	79346	1417	376	3,26	39,53
140	78349	1684	267	1,96	46,97
150	77860	2108	424	1,32	58,80
160	77104	2644	536	0,34	73,75
170	76883	3114	470	0,05	86,86
171	76875	3253	139	0,04	90,74
172	76854	3529	276	0,02	98,44
173	76842	3575	46	0,00	99,72
174	76842	3585	10	0,00	100,00

A mutação populacional consiste em realizar a mutação *swap* em todos os indivíduos da população depois de um determinado número de gerações sucessivas sem melhoria ter sido atingido. Ainda nesta etapa, quando um indivíduo gerado é repetido ele sofre mutação novamente até sua seqüência ser única na população. Este procedimento é descrito a seguir.

Passo 1 : Se $f_{0i} = f_{0(i-1)}$, então c := c+1, caso contrário c := 0;

Passo 2 : Se $c = G_e$, então:

- Se $f_{0i} < f^*$, então $s^* := s_{0i}$ e $f^* := f_{0i}$, caso contrário $s_{0i} := s^*$ e todos os indivíduos da população sofrem mutação *swap* e c := 0;

Onde: f* : é o valor da melhor aptidão em todas as gerações já realizadas;

s* : é o melhor indivíduo em todas as gerações já realizadas;

f_{0i} : é o valor da melhor aptidão na população *i*;

s_{0i} : é o melhor indivíduo da população *i*;

c : é o número de gerações sucessivas sem melhoria de f_{0i};

i : é o número da *i-ésima* geração; e

G_e : número de gerações sucessivas sem melhoria.

O segundo passo armazena o melhor indivíduo de todas as gerações e faz com que ele sempre esteja na população antes da mutação de todos os indivíduos.

4.3.6. Estratégia Geracional

A estratégia geracional implementada no rAG é inspirada na estratégia *population overlaps*. Para evitar que existam indivíduos repetidos na população, um indivíduo só é aceito para ser incorporado na população se a sua estrutura não é repetida. A outra condição depende da aptidão do indivíduo. Se a aptidão do novo indivíduo f é melhor do que a pior aptidão da população f_p, então este indivíduo substitui o indivíduo de pior aptidão e o valor da pior aptidão é atualizado.

Nesta etapa o primeiro procedimento proposto, baseado no princípio da diversificação, para melhorar o desempenho do rAG foi implementado. A estratégia de só aceitar um indivíduo com aptidão melhor que a pior aptidão diminuía a diversidade da população e, por isso, foi implementada a possibilidade controlada por um parâmetro de um indivíduo com aptidão inferior a pior aptidão existente na população ser aceito, deste que tenha seqüência única. Este procedimento é descrito a seguir:

Passo 1 : Execução do operador *crossover*;

Passo 2 : Se $f < f_p$ e $s \neq s_i$ ($\forall$ i = 1, 2, ..., N), então: $s_p := s$ e atualiza f_p;

Passo 3 : Se $f \geq f_p$ e $s \neq s_i$ ($\forall$ i = 1, 2, ..., N), então:

- gera-se um número aleatório *r* entre 0 e 1:
- se $r < P_a$, então: $s_p := s$ e atualiza f_p;

Onde:

N : é o tamanho da população;

P_a : probabilidade de aceitar um indivíduo;

s : é um dos indivíduos gerado pelo operador *crossover*;

f : é a aptidão de s;

s_p : é o indivíduo de pior aptidão; e

f_p : é a aptidão de s_p.

4.3.7. Critério de Parada

O critério de parada utilizado foi o tempo de execução, devido o rAG ter uma característica de manter a diversidade. Em comparação ao número de gerações, o tempo de execução é mais adequado tanto para o planejamento dos experimentos como para a facilidade da implementação computacional.

4.3.8. Parametrização do rAG

Ruiz *et al.* (2006) realizou um projeto de experimentos para encontrar a melhor combinação entre componentes e valores para os parâmetros do seu AG, o resultado foi um desvio de 3,22% e 3,85% para a melhor e a pior combinação de operadores e valores dos parâmetros, respectivamente. Segundo os autores está diferença não era muito significativa, o que demonstrava a robustez do AG desenvolvido, i.e., a qualidade das soluções obtidas pelo AG eram pouco dependente dos valores dos parâmetros. Devido a dificuldade da calibração dos valores dos parâmetros e a possibilidade de conseguir uma melhoria pequena, optou-se por não acrescentar aos objetivos deste trabalho realizar uma calibração otimizada dos parâmetros do rAG. Considerou-se mais importante construir um bom projeto para o AG que lhe atribuísse robustez.

Os valores dos parâmetros do rAG foram determinados durante a implementação computacional e os primeiros experimentos. Durante os experimentos se percebeu que o rAG se comportava melhor para o CPFSP com o critério de desempenho sendo o tempo total de fluxo, com o tamanho da população e o parâmetro G_e maiores que os outros dois

problemas testados. Assim, foram usados dois conjuntos de valores para os parâmetros do rAG, apresentados a seguir.

Os parâmetros do rAG e seus respectivos valores para o CPFSP com o tempo total de fluxo sendo o critério de desempenho:

- Tamanho da população (N) : 75;
- Tamanho do sub-conjunto de seleção (d) : 3;
- Parâmetro da seleção por torneio (k) : 0,7;
- Taxa de *crossover* (P_c) : 0,70;
- Taxa de aceitação (P_a) : 0,30;
- Taxa de *crossover* elitista (P_{ce}): 0,30;
- Taxa de mutação (P_m) : 0,05; e
- Gerações de estagnação (G_e) : 50.

Os parâmetros do rAG e seus respectivos valores para o CPFSP e o PFSP com o *makespan* sendo o critério de desempenho:

- Tamanho da população (N) : 30;
- Tamanho do sub-conjunto de seleção (d) : 3;
- Parâmetro da seleção por torneio (k) : 0,7;
- Taxa de *crossover* (P_c) : 0,70;
- Taxa de aceitação (P_a) : 0,30;
- Taxa de *crossover* elitista (P_{ce}): 0,30;
- Taxa de mutação (P_m) : 0,05; e
- Gerações de estagnação (G_e) : 25.

4.3.9. Resumo do rAG

Um resumo do rAG pode ser dado da seguinte forma: primeiro é gerada uma população inicial totalmente aleatória de tamanho N; em seguida é verificado se há algum indivíduo com seqüência repetida, se houver, este indivíduo sofre mutação até sua seqüência ser a única da população; depois todos os indivíduos da população são avaliados, i.e., recebem sua aptidão; o procedimento *crossover* é executado N vezes: é gerado um número aleatório entre 0 e 1, se for menor ou igual a P_c é realizado o *crossover* OX, se for maior é realizado

o *crossover* elitista. Para cada um dos indivíduos gerados no processo de *crossover* são testadas as condições do procedimento de estratégia geracional; então é executado o procedimento de mutação, realizado i vezes, de 1 a N, onde é gerado um número aleatório entre 0 e 1, se for menor ou igual a P_m então o indivíduo i sofre mutação; depois é determinada a aptidão do melhor indivíduo, se ela for igual ou menor que a melhor aptidão da população anterior um contador *c* é acrescentado de uma unidade, caso contrário, o contador *c* recebe zero; depois do contador ter sido atualizado, se o seu valor atingir o valor de G_e, então a mutação populacional é executada e o contador *c* recebe zero. O tempo de execução decorrido é comparado com o tempo de execução estabelecido no critério de parada, se for menor, os procedimentos anteriores deste o *crossover* são realizados novamente, caso contrário, a execução do algoritmo é encerrada. Percebe-se que numa mesma população nunca existem dois indivíduos com a mesma seqüência. O pseudocódigo do rAG é apresentado na Figura 4.7.

```
Gera a população inicial
Identifica e corrige indivíduos repetidos
Avalia a população
Enquanto o critério de parada é falso faça
  Escolhe o tipo de crossover
    Executa a seleção
    Executa o operador de crossover
  Executa o operador de mutação
  Se c = Ge, então
    Executa a mutação populacional
Fim do enquanto
```

Figura 4.7 – Pseudocódigo do rAG.

Depois da descrição do rAG faz-se uma abordagem explicitando a diferença em relação aos outros AGs apresentados neste trabalho. Os AGs apresentados neste trabalho para o CPFSP foram o GAChen de Chen *et al.* (1996), o GASA de Shuster e Framinan (2003) e o GA_AA de Aldowaisan e Allahverdi (2003). Já para o PFSP foram o GAChen de Chen *et al.* (1995), o GAMIT de Murata *et al.* (1996), o GAReev de Reeves (1995) e o GA_RMA

de Ruiz *et al.* (2006). A Tabela 4.2 mostra as diferenças entre os outros AGs e o rAG. Na Tabela 4.2: a coluna um mostra o nome do AG; a coluna dois mostra se a inicialização foi eficiente e qual heurística utilizada eficientemente ou não; a coluna três mostra se foi utilizada hibridização e qual o método usado; a coluna quatro mostra que tipo de operador *crossover* foi utilizado; a coluna cinco mostra se foi utilizado o operador de mutação com que procedimento; a coluna seis mostra a taxa de *crossover* utilizada; e a coluna sete mostra a taxa de mutação utilizada.

A análise da tabela 4.2 mostra que o rAG foi o único a não usar inicialização eficiente e/ou hibridização. Nota-se que quando os outros AG não têm inicialização eficiente, usam hibridização, isso é uma forma de melhorar a qualidade das soluções obtidas já que as soluções iniciais aleatórias são de baixa qualidade, ou quando tem inicialização eficiente não usam hibridização. Só um o GA_AA usou inicialização eficiente e hibridização ao mesmo tempo. Metade dos AGs comparados usou ao menos o valor de 100% para alguma das taxas dos operadores genéticos. Os resultados dos experimentos serão apresentados no próximo capítulo e mostrará se o rAG consegue ser competitivo mesmo sem usar inicialização eficiente ou hibridização.

Tabela 4.2 – Diferença entre os AG apresentados para o PFSP e o rAG.

AG	**Inic. eficiente**	**Hibridização**	***Crossover***	**Mutação**	**Taxa *Crossover***	**Taxa Mutação**
GAChen (CPFSP)	CDS e Dannenbring	Não	PMX	*Swap*	0,95	0,01
GASA	Não	Simulated annealing	DPA*	DPA*	1,00	1,00
GA_AA	CDS e Dannenbring	Busca local	PMX	*Swap*	0,95	0,01
GAChen (PFSP)	CDS e Dannenbring	Não	PMX	Não	1,00	0
GAMIT	Não	Busca local	*Two-point*	*Shift*	1,00	1,00
GA_Reev	NEH	Não	*One-point*	*Shift*	1,00	0,80
GA_RMA	NEH	Não	SBOX	*Shift*	0,40	0,01
rAG	Não	Não	OX	*Swap*	0,70	0,05

* Desenvolvido pelo próprio autor.

CAPÍTULO 5 – EXPERIMENTOS COMPUTACIONAIS

Os resultados dos experimentos computacionais realizados com o rAG e as comparações com os outros métodos são apresentados neste capítulo. O código do rAG foi implementado em Delphi 7. Os experimentos foram realizados em um computador PC-AMD (2.2 GHz e 512 MB de RAM). Devido a natureza probabilística dos AGs tradicionais, o rAG foi executado cinco vezes para cada problema e escolhido o melhor resultado.

O principal indicador utilizado nas comparações entre os métodos é o percentual de desvio das soluções, dado por ((s* - s') / s*)x100, onde s* é a melhor solução do problema e s' é a melhor solução encontrada pelo método de resolução aplicado ao problema.

Um conjunto de experimentos foi programado para ser realizado e os resultados obtidos são apresentados nas seções 5.1 a 5.5, dados adiante.

Além disso, é apresentado um conjunto de polinômios do segundo grau construídos a partir dos resultados obtidos com o rAG aplicado ao PFSP, com o objetivo de determinar *a priori* o tempo de execução necessário a ser gasto na aplicação do método a partir da qualidade da solução desejada.

Por fim, descrevemos como foi o desempenho do rAG aplicado num caso real.

5.1. Experimento 1 – Etapas de melhoria do rAG

O primeiro experimento tem o objetivo de demonstrar e analisar a melhoria obtida pelo rAG com a utilização dos procedimentos propostos baseados nos princípios da diversidade e intensificação no CPFSP. Nesta etapa foram realizados 4 tipos de experimentos. O primeiro experimento foi realizado com o chamado rAG-1 que não tem implementado nenhum dos três procedimentos propostos. O segundo experimento foi realizado com o rAG-1 acrescido do procedimento que permite que indivíduos com aptidão menor que a pior aptidão da população tenham probabilidade de serem aceitos na população, e foi denominado de rAG-2. O terceiro experimento foi realizado com o rAG-2 acrescido do

procedimento *crossover* elitista, e foi denominado de rAG-3. O quarto e último experimento nesta etapa foi realizado com o rAG-3 acrescido do procedimento mutação populacional, e foi denominado de rAG-4. Todos os experimentos foram realizados com as instâncias de Taillard (1993).

O resumo dos resultados desses experimentos com os tempos de execução usados são mostrados na Tabela 5.1. Nesta tabela a coluna um mostra as classes das instâncias de Taillard (1993), as colunas dois a cinco mostram o percentual de desvio para o rAG-1, rAG-2, rAG-3 e rAG-4, respectivamente, e a coluna seis mostra o tempo de execução usado em cada classe de problemas.

A análise dos dados da Tabela 5.1 produz as seguintes observações:

a) O primeiro procedimento (rAG-2) passou o desvio do rAG de 1,710% para 0,517%, o segundo procedimento (rAG-3) passou o desvio de 0,517% para 0,174% e o terceiro procedimento (rAG-4) passou o desvio de 0,174% para 0,171%;
b) O acréscimo do último procedimento melhorou o desempenho do rAG em apenas 0,003%, isto se explica pelos baixos tempos de execução utilizados, pois a característica deste procedimento é agir quando a população se encontra em estado de estagnação e o que não ocorre com poucas gerações executadas;
c) A eficiência do terceiro procedimento é sentida nas instâncias com 20 tarefas porque estes problemas são menos complexos e, por isso, rapidamente o rAG encontra boas soluções e, por isso, a população entra em estado de estagnação e a mutação populacional passa a ter um papel ativo; e
d) O resultado do rAG-4 ter sido inferior ao resultado do rAG-3 para a classe 100x5 se deve a utilização dos números aleatórios gerados pelo Delphi 7, e como esses dois algoritmos têm quantidades de execuções de números aleatórios diferentes, isso influenciou no resultado.

Estes resultados mostram que foi proveitosa a implementação dos procedimentos propostos para o desempenho do rAG. O rAG-4 foi o que obteve os melhores resultados, por isso, passará a ser chamado apenas de rAG e será o algoritmo usado daqui para frente.

Tabela 5.1 – Resumo da comparação das várias etapas de melhoria do rAG.

Instâncias	rAG-1	rAG-2	rAG-3	rAG-4	Tempo (s)
20 x 5	0,55	0,15	0,12	0,04	0,08
20 x 10	0,39	0,11	0,04	0,02	0,08
20 x 20	0,51	0,11	0,08	0,05	0,08
50 x 5	0,67	0,22	0,15	0,14	3,75
50 x 10	0,70	0,29	0,18	0,15	3,75
50 x 20	1,01	0,40	0,21	0,12	3,75
100 x 5	1,83	0,59	-0,10	-0,01	10,00
100 x 10	2,68	0,71	0,18	0,18	10,00
100 x 20	2,86	1,03	0,51	0,34	10,00
200 x 10	3,28	0,79	0,00	0,16	50,00
200 x 20	4,33	1,28	0,53	0,68	50,00
Média	**1,710**	**0,517**	**0,174**	**0,171**	**15,96**

As Tabelas 5.2 a 5.5 mostram os resultados dos quatro métodos avaliados para cada um dos 110 problemas de Taillard (1993). A especificação de cada coluna dessas tabelas é a seguinte: a coluna um mostra a descrição dos problemas, a coluna dois mostra o melhor resultado obtido por Fink e Voβ (2003) referenciado como FV, a coluna três mostra o resultado obtido pelo rAG-1, a coluna quatro mostra desvio do rAG-1 em relação a FV, a coluna cinco mostra o resultado obtido pelo rAG-2, a coluna seis mostra o desvio do rAG-2 em relação a FV, a coluna sete mostra o resultado obtido pelo rAG-3, a coluna oito mostra o desvio percentual do rAG-3 em relação a FV, a coluna nove mostra o resultado obtido pelo rAG-4 e a coluna dez mostra o desvio percentual do rAG-4 em relação a FV.

A análise das informações contidas nas Tabelas 5.2 a 5.5 trás as seguintes observações:

a) Na Tabela 5.2 sabendo-se que as soluções FV para as classes com 20 tarefas são as soluções ótimas dos problemas, o rAG-1 só obteve 3 soluções ótimas, o rAG-2 obteve 13 soluções ótimas, o rAG-3 alcançou 19 soluções ótimas e o rAG-4 obteve 24 soluções ótimas;
b) Na Tabela 5.3 para as classes com 50 tarefas o rAG-1 não obteve nenhuma solução melhor que FV, já o rAG-2 conseguiu 5 soluções melhores que FV, o rAG-3 e o rAG-4 obtiveram, cada um, 7 soluções melhores que FV;
c) Na Tabela 5.4 para as classes com 100 tarefas o rAG-1 e o rAG-2 não obtiveram nenhuma solução melhor que FV, já o rAG-3 obteve 7 soluções melhores que FV e o rAG-4 também obteve 7 soluções melhores que FV; e

d) Na Tabela 5.5 Para as classes com 200 tarefas o rAG-1 não obteve nenhuma solução melhor que FV, já o rAG-2 conseguiu 1 solução melhor que FV, o rAG-3 obteve 6 soluções melhores que FV e o rAG-4 obteve 4 soluções melhores que FV.

Estes resultados demonstram mais uma vez que as incorporações dos procedimentos propostos ao rAG melhoram cada vez mais a capacidade deste algoritmo de obter soluções melhores. Prova disso foi a mutação populacional que fez o algoritmo passar de 19 soluções ótimas obtidas pelo rAG-3 para as classes com 20 tarefas para 24 soluções ótimas obtidas.

Tabela 5.2 – Comparação das várias etapas de melhoria do rAG para as classes com n = 20.

Instâncias	FV	rAG-1	D (%)	rAG-2	D (%)	rAG-3	D (%)	rAG-4	D (%)
20x5									
tai001	15674	15698	0,15	15674	0,00	15674	0,00	15674	0,00
tai002	17250	17368	0,68	17297	0,27	17250	0,00	17250	0,00
tai003	15821	16062	1,52	15969	0,94	15821	0,00	15877	0,35
tai004	17970	18043	0,41	17999	0,16	17970	0,00	17970	0,00
tai005	15317	15490	1,13	15331	0,09	15317	0,00	15317	0,00
tai006	15501	15527	0,17	15501	0,00	15673	1,11	15501	0,00
tai007	15693	15767	0,47	15696	0,02	15706	0,08	15693	0,00
tai008	15955	16015	0,38	15965	0,06	15957	0,01	15963	0,05
tai009	16385	16394	0,05	16385	0,00	16385	0,00	16385	0,00
tai010	15329	15413	0,55	15329	0,00	15329	0,00	15329	0,00
Média			**0,55**		**0,15**		**0,12**		**0,04**
20x10									
tai011	25205	25417	0,84	25290	0,34	25206	0,00	25205	0,00
tai012	26342	26540	0,75	26342	0,00	26342	0,00	26342	0,00
tai013	22910	22936	0,11	22936	0,11	22910	0,00	22910	0,00
tai014	22243	22315	0,32	22243	0,00	22243	0,00	22243	0,00
tai015	23150	23191	0,18	23191	0,18	23150	0,00	23150	0,00
tai016	22011	22179	0,76	22011	0,00	22048	0,17	22011	0,00
tai017	21939	21965	0,12	21939	0,00	21939	0,00	21939	0,00
tai018	24158	24158	0,00	24158	0,00	24205	0,19	24205	0,19
tai019	23501	23652	0,64	23503	0,01	23501	0,00	23501	0,00
tai020	24597	24633	0,15	24699	0,41	24597	0,00	24597	0,00
Média			**0,39**		**0,11**		**0,04**		**0,02**
20x20									
tai021	38597	38597	0,00	38597	0,00	38597	0,00	38597	0,00
tai022	37571	37798	0,60	37643	0,19	37686	0,31	37571	0,00
tai023	38312	38530	0,57	38312	0,00	38312	0,00	38382	0,18
tai024	38802	39264	1,19	38802	0,00	38812	0,03	38802	0,00
tai025	39012	39296	0,73	39096	0,22	39012	0,00	39073	0,16
tai026	38562	38808	0,64	38620	0,15	38618	0,15	38562	0,00
tai027	39663	39697	0,09	39738	0,19	39730	0,17	39744	0,20
tai028	37000	37342	0,92	37027	0,07	37027	0,07	37000	0,00
tai029	39228	39228	0,00	39228	0,00	39267	0,10	39228	0,00
tai030	37931	38076	0,38	38024	0,25	37931	0,00	37931	0,00
Média			**0,51**		**0,11**		**0,08**		**0,05**

Tabela 5.3 – Comparação das várias etapas de melhoria do rAG para as classes com n = 50.

Instâncias	FV	rAG-1	D (%)	rAG-2	D (%)	rAG-3	D (%)	rAG-4	D (%)
50x5									
tai031	76016	76318	0,40	75801	-0,28	75883	-0,17	75882	-0,18
tai032	83403	83733	0,40	83348	-0,07	83589	0,22	83240	-0,20
tai033	78282	78675	0,50	78732	0,57	78747	0,59	78467	0,24
tai034	82737	83059	0,39	82716	-0,03	82517	-0,27	82639	-0,12
tai035	83901	85400	1,79	84353	0,54	84338	0,52	83980	0,09
tai036	80924	81272	0,43	81319	0,49	80969	0,06	81471	0,68
tai037	78791	79071	0,36	79161	0,47	79172	0,48	79051	0,33
tai038	79007	79693	0,87	79220	0,27	79067	0,08	79263	0,32
tai039	75842	76308	0,61	76054	0,28	75892	0,07	75951	0,14
tai040	83829	84669	1,00	83787	-0,05	83785	-0,05	83882	0,06
Média			**0,67**		**0,22**		**0,15**		**0,14**
50x10									
tai041	114398	115267	0,76	114552	0,13	114473	0,07	114412	0,01
tai042	112725	112988	0,23	112853	0,11	112821	0,09	112408	-0,28
tai043	105433	105890	0,43	105706	0,26	105786	0,33	105694	0,25
tai044	113540	113972	0,38	113809	0,24	113932	0,35	113994	0,40
tai045	115441	116552	0,96	115695	0,22	115896	0,39	115434	-0,01
tai046	112645	113080	0,39	113951	1,16	112522	-0,11	112904	0,23
tai047	116560	117339	0,67	116896	0,29	116890	0,28	116809	0,21
tai048	115056	115865	0,70	115042	-0,01	114995	-0,05	115166	0,10
tai049	110482	111906	1,29	110629	0,13	110814	0,30	110510	0,03
tai050	113462	114775	1,16	113922	0,41	113670	0,18	114145	0,60
Média			**0,70**		**0,29**		**0,18**		**0,15**
50x20									
tai51	172845	173773	0,54	173856	0,58	173402	0,32	173154	0,18
tai52	161092	162001	0,56	161867	0,48	161442	0,22	161260	0,10
tai53	160213	162548	1,46	160891	0,42	160161	-0,03	160625	0,26
tai54	161557	163041	0,92	162217	0,41	161883	0,20	162382	0,51
tai55	167640	169667	1,21	167750	0,07	167397	-0,14	167140	-0,30
tai56	161784	163228	0,89	162525	0,46	162507	0,45	161939	0,10
tai57	167233	169459	1,33	167859	0,37	167291	0,03	167271	0,02
tai58	168100	170213	1,26	169047	0,56	168820	0,43	167822	-0,17
tai59	165292	167003	1,04	165895	0,36	165642	0,21	165292	0,00
tai60	168386	169912	0,91	168935	0,33	169148	0,45	169144	0,45
Média			**1,01**		**0,40**		**0,21**		**0,12**

Tabela 5.4 – Comparação das várias etapas de melhoria do rAG para as classes com n = 100.

Instâncias	FV	rAG-1	D (%)	rAG-2	D (%)	rAG-3	D (%)	rAG-4	D (%)
100x5									
tai061	308052	315572	2,44	309755	0,55	309589	0,50	307853	-0,06
tai062	302386	307787	1,79	303039	0,22	300877	-0,50	300301	-0,69
tai063	295239	296411	0,40	295705	0,16	293781	-0,49	294746	-0,17
tai064	278811	283647	1,73	283291	1,61	279040	0,08	280520	0,61
tai065	292757	299463	2,29	294944	0,75	293554	0,27	293529	0,26
tai066	290819	297639	2,35	291495	0,23	289502	-0,45	291328	0,18
tai067	300068	307786	2,57	302814	0,92	300934	0,29	302754	0,90
tai068	291859	295408	1,22	292362	0,17	290175	-0,58	289477	-0,82
tai069	307650	312164	1,47	309988	0,76	307174	-0,15	307874	0,07
tai070	301942	308223	2,08	303687	0,58	302161	0,07	300921	-0,34
Média			**1,83**		**0,59**		**-0,10**		**-0,01**
100x10									
tai071	412700	425317	3,06	415571	0,70	413587	0,21	410052	-0,64
tai072	394562	404554	2,53	394619	0,01	396092	0,39	393033	-0,39
tai073	405878	416555	2,63	406606	0,18	406166	0,07	407103	0,30
tai074	422301	434956	3,00	424781	0,59	422966	0,16	422508	0,05
tai075	400175	410467	2,57	405725	1,39	401531	0,34	402891	0,68
tai076	391359	398699	1,88	393526	0,55	392833	0,38	391755	0,10
tai077	394179	405388	2,84	397054	0,73	393455	-0,18	398792	1,17
tai078	402025	415212	3,28	405845	0,95	402134	0,03	402541	0,13
tai079	416833	426680	2,36	420429	0,86	416521	-0,07	417296	0,11
tai080	410372	421285	2,66	414894	1,10	412277	0,46	411741	0,33
Média			**2,68**		**0,71**		**0,18**		**0,18**
100x20									
tai081	562150	578291	2,87	568276	1,09	566299	0,74	563576	0,25
tai082	563923	576738	2,27	572882	1,59	567295	0,60	566027	0,37
tai083	562404	579826	3,10	570003	1,35	564884	0,44	565168	0,49
tai084	562918	582030	3,40	568895	1,06	566140	0,57	564664	0,31
tai085	556311	569888	2,44	560124	0,69	558504	0,39	557451	0,20
tai086	562253	582624	3,62	567371	0,91	565028	0,49	566607	0,77
tai087	574102	589156	2,62	575352	0,22	575047	0,16	574813	0,12
tai088	578119	595341	2,98	586900	1,52	581522	0,59	580907	0,48
tai089	564803	581740	3,00	570992	1,10	568157	0,59	565182	0,07
tai090	572798	585800	2,27	577125	0,76	575738	0,51	574652	0,32
Média			**2,86**		**1,03**		**0,51**		**0,34**

Tabela 5.5 – Comparação das várias etapas de melhoria do rAG para as classes com n = 200.

Instâncias	FV	rAG-1	D (%)	rAG-2	D (%)	rAG-3	D (%)	rAG-4	D (%)
200x10									
tai091	1521201	1575895	3,60	1531594	0,68	1527482	0,41	1526609	0,36
tai092	1516009	1559616	2,88	1529300	0,88	1506553	-0,62	1512508	-0,23
tai093	1515535	1580957	4,32	1539853	1,60	1522691	0,47	1515614	0,01
tai094	1489457	1533010	2,92	1508382	1,27	1500234	0,72	1499035	0,64
tai095	1513281	1561600	3,19	1508947	-0,29	1510513	-0,18	1511402	-0,12
tai096	1508331	1547496	2,60	1514446	0,41	1508429	0,01	1513766	0,36
tai097	1541419	1593727	3,39	1553749	0,80	1538915	-0,16	1554570	0,85
tai098	1533397	1578312	2,93	1546345	0,84	1526500	-0,45	1526915	-0,42
tai099	1507422	1559530	3,46	1519648	0,81	1507978	0,04	1504755	-0,18
tai100	1520800	1573829	3,49	1534494	0,90	1517887	-0,19	1525728	0,32
Média			**3,28**		**0,79**		**0,00**		**0,16**
200x20									
tai101	2012785	2104986	4,58	2041445	1,42	2032113	0,96	2032802	0,99
tai102	2057409	2143591	4,19	2081288	1,16	2055025	-0,12	2070359	0,63
tai103	2050169	2128363	3,81	2072699	1,10	2053455	0,16	2057855	0,37
tai104	2040946	2128780	4,30	2075877	1,71	2048040	0,35	2053048	0,59
tai105	2027138	2112664	4,22	2062773	1,76	2041657	0,72	2047943	1,03
tai106	2046542	2132140	4,18	2064979	0,90	2059528	0,63	2057600	0,54
tai107	2045906	2149991	5,09	2072585	1,30	2061357	0,76	2063947	0,88
tai108	2044218	2130926	4,24	2070443	1,28	2061633	0,85	2055332	0,54
tai109	2037040	2129088	4,52	2050343	0,65	2038011	0,05	2042594	0,27
tai110	2046966	2132272	4,17	2077830	1,51	2066081	0,93	2066217	0,94
Média			**4,33**		**1,28**		**0,53**		**0,68**

Aproveitando os resultados do rAG-4 construiu-se a Tabela 5.6 para verificar se depois de todos os processos de melhoria o rAG tinha se tornado tão eficiente quanto a melhor heurística para o CPFSP com o critério de desempenho sendo o tempo total de fluxo.

Tabela 5.6 – Comparação do rAG com a heurística Pilot-10-Chins.

Instâncias	Desvio (%)			Tempo (s)		
(*n*)	Pilot-10-Chins	rAG	DIF_{desvio}	Pilot-10-Chins	rAG	DIF_{tempo}
20	0,27	0,04	-0,23	0,8	0,08	0,80
50	0,85	0,14	-0,71	37,5	3,75	0,80
100	1,30	0,17	-1,13	879,0	10,00	0,09
200	0,57	0,42	-0,15	7.612,4	50,00	0,05
Média	**0,75**	**0,19**	**-0,56**	**2.132,4**	**15,96**	**0,44**

A Tabela 5.6, dada acima, mostra o resumo dos resultados do rAG e a comparação com a heurística Pilot-10-Chins. Nesta tabela a coluna um mostra o número de tarefas das instâncias, cada linha representa a agregação dos problemas com 5, 10 e 20 máquinas, a coluna dois mostra a média do desvio da heurística Pilot-10-Chins em relação as melhores soluções encontradas por FV, a coluna três mostra o desvio dos resultados do rAG em relação a FV, a coluna quatro (DIF_{desvio}) mostra a diferença entre os desvios do rAG e da heurística Pilo-10-Chins, a coluna cinco mostra os tempos de execução usados pela heurística Pilot-10-Chins, a coluna seis mostra os tempos de execução usados pelo rAG, a coluna sete (DIF_{tempo}) mostra a razão entre o tempo de execução usado pela heurística Pilot-10-Chins e pelo rAG já multiplicado por 8 que é a razão entre as velocidades dos dois computadores (2.200/266).

Sobre a Tabela 5.6 descrevemos as seguintes observações.

a) Nas quatro classes de problemas o rAG foi superior a heurística Pilot-10-Chins. Na média o rAG foi 0,56 % superior a heurística Pilot-10-Chins;
b) O tempo de execução do rAG para as quatro classes de problemas sempre foi inferior ao utilizado pela heurística Pilot-10-Chins. A média da razão entre os tempos usados pela heurística Pilot-10-Chins e pelo rAG foi 0,44 o que significa que o rAG foi mais rápido que a heurística Pilot-10-Chins;
c) A menor razão entre os tempos usados pela heurística Pilot-10-Chins e pelo rAG foi 0,05 e a maior razão foi 0,80;
d) O melhor resultado do rAG em comparação com a heurísta Pilot-10-Chins foi -1,13% na classe com 100 tarefas; e
e) A menor diferença entre o rAG em comparação com a heurística Pilot-10-Chins foi -0,15% na classe com 200 tarefas.

Estes resultados demonstram que o rAG é mais eficiente e eficaz que a heurística Pilot-10-Chins, o que contradiz a opinião de alguns autores (Reeves e Rowe, 2002; Dréo *et al.*, 2006) que afirmam que um AG que é uma técnica de busca global não consegue ser melhor que uma técnica que usa o conhecimento específico do problema para guiar a busca nas mesmas condições de tempo de execução.

5.2. Experimento 2 – rAG x FV

O objetivo deste segundo tipo de experimento é analisar a capacidade do rAG em obter soluções tão boas ou melhores que as soluções encontradas por Fink e Voβ (2003) para o CPFSP com o tempo total de fluxo como critério de desempenho. Os experimentos do FV foram realizados num computador Pentium II (266 MHz). Então a razão entre as velocidades dos dois computadores utilizados nos experimentos é 2.200/266, ou seja, 8,27. Por isso, os tempos de execução do rAG foram adotados para serem menores que esta razão. Os problemas de testes utilizados foram às instâncias de Taillard (1993). O resumo dos resultados do rAG e a comparação com o FV são mostrados na Tabela 5.7. Na Tabela 5.7 a coluna um mostra as classes de problemas das instâncias de Taillard (1993), a coluna dois mostra os tempos usados pelo FV, a coluna três mostra os tempos usados pelo rAG, a coluna quatro mostra a razão do tempo usado pelo rAG e pelo FV já multiplicado por 8 que é a diferença entre as velocidades dos computadores utilizados nos testes e a coluna cinco mostra o desvio percentual do rAG em relação ao FV.

Como o CPFSP é o problema principal deste trabalho e as instâncias de Taillard (1993) são as mais utilizadas nas comparações entre os métodos propostos para os problemas da classe FSP, decidiu-se fazer uma análise mais profunda nos experimentos realizados. Para isso, foram criados alguns indicadores que são definidos a seguir. Os resultados para os testes com os 110 problemas de Taillard (1993) com os valores dos indicadores são apresentados nas Tabelas 5.8 a 5.11.

a) FV : é o melhor resultado obtido por Fink e Voβ (2003);
b) rAG* : é o melhor resultado obtido pelo rAG nas cinco execuções de cada problema;
c) rAG_P : é o pior resultado obtido pelo rAG nas cinco execuções de cada problema;
d) D (rAG_P e rAG*) : é o desvio percentual entre o pior e o melhor resultado obtido pelo rAG e calculado desta forma: $((rAG^* - rAG_p)/rAG^*)x100$;
e) G_T : é o total de gerações executadas na obtenção de rAG*;
f) G* : é a geração na qual foi obtido o valor rAG*;
g) t* : é o tempo decorrido até obter rAG*; e

h) D (rAG* e FV) : é o desvio percentual entre o valor rAG* e o valor FV, calculado desta forma: ((FV – rAG*)/FV)x100.

A análise da tabela 5.7 produz os seguintes resultados.

a) Nas classes com 20 tarefas onde as soluções de FV são ótimas o rAG também obteve todas as soluções ótimas;
b) Nas instâncias restantes o rAG foi sempre melhor que os resultados de FV. Na média o rAG foi 0,34 % superior ao FV;
c) Os tempos de execução do rAG foram sempre inferiores aos tempos usados pelo FV considerando a conversão. A razão entre os tempos de execução do rAG e do FV foi 0,18 o que significa que o rAG é mais rápido que o FV;
d) A menor razão entre os tempos usados pelo rAG e o FV foi 0,006 o que significa que o rAG foi 166 vezes mais rápido que o FV e a maior razão foi 0,463 o que significa que o rAG foi 2 vezes mais rápido que o FV;
e) O melhor resultado do rAG em comparação com o FV foi -1,13 % na classe 200x10; e
f) Quando o rAG foi melhor que o FV a menor diferença foi -0,01 % na classe 50x20.

Estes resultados mostram claramente que o rAG é ao mesmo tempo mais eficaz e eficiente que o FV. É mais eficaz porque obtem os melhores resultados e mais eficiente porque faz isso em menos tempo.

Tabela 5.7 – Resumo da comparação do rAG com o FV.

Instância (n x m)	Tempo (s)			Desvio (FV e rAG*) (%)
	FV	rAG*	(rAG* / FV) x 8	
20 x 5	1.000,80	0,75	0,006	0,00
20 x 10	1.000,80	1,50	0,012	0,00
20 x 20	1.000,80	3,00	0,024	0,00
50 x 5	1.037,50	15,00	0,116	-0,14
50 x 10	1.037,50	30,00	0,231	-0,11
50 x 20	1.037,50	60,00	0,463	-0,01
100 x 5	1.879,00	25,00	0,106	-0,53
100 x 10	1.879,00	50,00	0,213	-0,56
100 x 20	1.879,00	100,00	0,426	-0,46
200 x 10	8.612,40	150,00	0,139	-1,13
200 x 20	8.612,40	300,00	0,279	-0,82
Média	**2.634,25**	**66,84**	**0,18**	**-0,34**

A análise das Tabelas 5.8 a 5.11 produz os seguintes resultados.

a) Na Tabela 5.8 oito problemas das classes com 20 tarefas o rAG obteve a mesma solução nas cinco execuções (tai005, tai017, tai022, tai024, tai027, tai028, tai029 e tai030), sendo que estas são as soluções ótimas dos problemas. Também no problema tai053 da classe 50x20 o rAG obteve a mesma solução nas cinco execuções, sendo que a solução é melhor que a solução de FV;
b) A maior diferença entre a melhor solução encontrada pelo rAG e a pior solução foi de 2,66 % no problema tai006;
c) A menor média entre a diferença da melhor e pior solução encontrada pelo rAG foi de 0,09 % para a classe 20x20;
d) A maior média entre a diferença da melhor e pior solução encontrada pelo rAG foi de 0,99 % para a classe 100x10;
e) Considerando apenas os 80 problemas que não tem solução ótima definida que são os problemas com 50, 100 e 200 tarefas o rAG obteve 69 soluções melhores que o FV, 1 solução igual e 10 soluções inferiores ao FV. Nove problemas que o FV foi melhor são das classes com 50 tarefas e um da classe com 100 tarefas. Todos os problemas das classes com 200 tarefas o rAG foi melhor que o FV. Isto mostra uma tendência do rAG ser melhor quando o número de tarefas aumenta;
f) A melhor solução mais rápida obtida pelo rAG foi no problema tai008 com o tempo de 0,02s em 29 gerações;

g) A maior diferença entre os desvios percentuais do rAG e do FV favorável ao rAG foi de -1,68 % para o problema tai098;
h) A maior diferença entre os desvios percentuais do rAG e do FV favorável ao FV foi de 0,22 % para o problema tai058;

A análise dos resultados dos indicadores mostra a influência das cinco execuções para cada problema nos resultados do rAG. As cinco execuções são justificadas pela natureza aleatória do AG. Mesmo se o tempo de execução fosse contado como o tempo das cinco execuções o rAG ainda estaria usando tempo de execução equivalente ao FV já que atualmente cada execução usa em média 20% do tempo de FV e passaria a 100%. Mesmo considerando este detalhe o rAG ainda é um método melhor que o FV porque consegue obter soluções melhores.

Tabela 5.8 – Resultados da comparação do rAG com o FV para a classe n = 20.

Instâncias	FV	rAG*	rAG_P	D (rAG_P e rAG*) %	G_T	G*	t* (s)	D (rAG* e FV) %
20x5								
tai001	15674	15674	15754	0,51	1159	250	0,16	0,00
tai002	17250	17250	17393	0,82	1266	197	0,12	0,00
tai003	15821	15821	15886	0,41	1312	473	0,27	0,00
tai004	17970	17970	18026	0,31	1275	212	0,12	0,00
tai005	15317	15317	15317	0,00	1265	117	0,07	0,00
tai006	15501	15501	15924	2,66	1241	103	0,06	0,00
tai007	15693	15693	15789	0,61	1293	424	0,25	0,00
tai008	15955	15955	15968	0,08	1273	29	0,02	0,00
tai009	16385	16385	16489	0,63	1252	915	0,55	0,00
tai010	15329	15329	15486	1,01	1292	1192	0,69	0,00
Média				**0,70**			**0,23**	**0,00**
20x10								
tai011	25205	25206	25292	0,34	2412	2252	1,40	0,00
tai012	26342	26342	26388	0,17	2460	1010	0,62	0,00
tai013	22910	22910	23043	0,58	2442	82	0,05	0,00
tai014	22243	22243	22314	0,32	2576	183	0,11	0,00
tai015	23150	23150	23269	0,51	2427	778	0,48	0,00
tai016	22011	22011	22185	0,78	2466	958	0,58	0,00
tai017	21939	21939	21939	0,00	2435	198	0,12	0,00
tai018	24158	24158	24205	0,19	2502	303	0,18	0,00
tai019	23501	23501	23651	0,63	2496	37	0,02	0,00
tai020	24597	24597	24715	0,48	2380	69	0,04	0,00
Média				**0,40**			**0,36**	**0,00**
20x20								
tai021	38597	38597	38855	0,66	4970	1673	1,01	0,00
tai022	37571	37571	37571	0,00	4931	101	0,06	0,00
tai023	38312	38312	38337	0,07	4973	444	0,27	0,00
tai024	38802	38802	38802	0,00	5161	571	0,33	0,00
tai025	39012	39012	39038	0,07	5033	160	0,10	0,00
tai026	38562	38562	38612	0,13	4845	943	0,58	0,00
tai027	39663	39663	39663	0,00	5034	1919	1,14	0,00
tai028	37000	37000	37000	0,00	5034	3634	2,17	0,00
tai029	39228	39228	39228	0,00	4850	543	0,34	0,00
tai030	37931	37931	37931	0,00	4990	3883	2,33	0,00
Média				**0,09**			**0,83**	**0,00**

Tabela 5.9 – Resultados da comparação do rAG com o FV para a classe n = 50.

Instâncias	FV	rAG*	rAG_P	D (rAG_P e rAG*) %	G_T	G*	t* (s)	D (rAG* e FV) %
50x5								
tai031	76016	76148	76492	0,45	13078	2171	2,49	0,17
tai032	83403	83172	83566	0,47	12552	7851	9,38	-0,28
tai033	78282	78416	79621	1,51	13030	10776	12,41	0,17
tai034	82737	82483	83185	0,84	12876	11330	13,20	-0,31
tai035	83901	83514	84280	0,91	12568	7692	9,18	-0,46
tai036	80924	80763	81523	0,93	13061	7671	8,81	-0,20
tai037	78791	78669	78964	0,37	12979	9030	10,44	-0,15
tai038	79007	79046	79449	0,51	13068	9154	10,51	0,05
tai039	75842	75830	76502	0,88	13190	13101	14,90	-0,02
tai040	83829	83550	84710	1,37	13161	3755	4,28	-0,33
Média				**0,82**			**9,56**	**-0,14**
50x10								
tai041	114398	114177	114731	0,48	26516	7777	8,80	-0,19
tai042	112725	112116	113429	1,16	25804	24534	28,52	-0,54
tai043	105433	105345	105854	0,48	26238	19988	22,85	-0,08
tai044	113540	113387	113733	0,30	25973	24568	28,38	-0,13
tai045	115441	115425	115781	0,31	26163	23082	26,47	-0,01
tai046	112645	112489	113343	0,75	25859	8631	10,01	-0,14
tai047	116560	116617	117272	0,56	25994	20307	23,44	0,05
tai048	115056	115097	116042	0,81	26110	16205	18,62	0,04
tai049	110482	110451	111184	0,66	24991	4856	5,83	-0,03
tai050	113462	113427	113792	0,32	26017	13813	15,93	-0,03
Média				**0,58**			**18,88**	**-0,11**
50x20								
tai051	172845	172740	174301	0,90	53980	27555	30,63	-0,06
tai052	161092	160980	161517	0,33	53904	15726	17,50	-0,07
tai053	160213	160104	160104	0,00	53284	22133	24,92	-0,07
tai054	161557	161678	162382	0,43	53072	33605	37,99	0,07
tai055	167640	167081	167410	0,20	50458	18719	22,26	-0,33
tai056	161784	162027	162347	0,20	53452	28936	32,48	0,15
tai057	167233	167098	167658	0,33	54438	19991	22,03	-0,08
tai058	168100	168462	168828	0,22	53174	38976	43,98	0,22
tai059	165292	165292	167012	1,03	52896	50384	57,15	0,00
tai060	168386	168560	169643	0,64	54052	32681	36,28	0,10
Média				**0,43**			**32,52**	**-0,01**

Tabela 5.10 – Resultados da comparação do rAG com o FV para a classe n = 100.

Instâncias	FV	rAG*	rAG_P	D (rAG_P e rAG*) %	G_T	G*	t* (s)	D (rAG* e FV) %
100x5								
tai061	308052	307329	309689	0,76	12290	11653	23,70	-0,23
tai062	302386	299602	301866	0,75	12129	8416	17,35	-0,92
tai063	295239	292693	294230	0,52	12310	10891	22,12	-0,86
tai064	278811	278860	281998	1,11	12327	10887	22,08	0,02
tai065	292757	291803	293955	0,73	12269	5569	11,35	-0,33
tai066	290819	288486	291211	0,94	12010	10986	22,87	-0,80
tai067	300068	299353	303267	1,29	12059	10729	22,24	-0,24
tai068	291859	289262	293328	1,39	12176	11058	22,70	-0,89
tai069	307650	306259	307417	0,38	12137	10201	21,01	-0,45
tai070	301942	300154	304206	1,33	12326	12162	24,67	-0,59
Média				**0,92**			**21,01**	**-0,53**
100x10								
tai071	412700	410289	415227	1,19	24472	17599	35,96	-0,58
tai072	394562	392032	395106	0,78	24769	22584	45,59	-0,64
tai073	405878	403399	405472	0,51	24437	20542	42,03	-0,61
tai074	422301	419582	422681	0,73	24125	14239	29,51	-0,64
tai075	400175	399019	400056	0,26	24406	24096	49,36	-0,29
tai076	391359	389123	393616	1,14	24883	19959	40,11	-0,57
tai077	394179	393034	397527	1,13	24431	21610	44,23	-0,29
tai078	402025	398617	403488	1,21	24302	23752	48,87	-0,85
tai079	416833	414093	418748	1,11	24625	20556	41,74	-0,66
tai080	410372	408580	416365	1,87	24111	20787	43,11	-0,44
Média				**0,99**			**42,05**	**-0,56**
100x20								
tai081	562150	559459	562639	0,57	50026	45144	90,24	-0,48
tai082	563923	563649	568368	0,83	49717	47577	95,70	-0,05
tai083	562404	560260	563821	0,63	49708	42297	85,09	-0,38
tai084	562918	561826	565257	0,61	48850	46424	95,03	-0,19
tai085	556311	552584	557465	0,88	49571	45582	91,95	-0,67
tai086	562253	560120	561869	0,31	48996	46362	94,62	-0,38
tai087	574102	570282	573831	0,62	48448	26977	55,68	-0,67
tai088	578119	575843	581755	1,02	49565	18913	38,16	-0,39
tai089	564803	562473	564726	0,40	49455	40550	81,99	-0,41
tai090	572798	567060	576671	1,67	48778	34967	71,69	-1,00
Média				**0,75**			**80,02**	**-0,46**

Tabela 5.11 – Resultados da comparação do rAG com o FV para a classe n = 200.

Instâncias	FV	rAG*	rAG_P	D (rAG_P e rAG*) %	G_T	G*	t* (s)	D (rAG* e FV) %
200x10								
tai091	1521201	1508293	1517798	0,63	38587	34374	133,62	-0,85
tai092	1516009	1498266	1506730	0,56	38437	32743	127,78	-1,17
tai093	1515535	1500991	1510431	0,62	38577	36208	140,79	-0,96
tai094	1489457	1477786	1500640	1,52	38674	37874	146,90	-0,78
tai095	1513281	1490851	1505283	0,96	38662	38649	149,95	-1,48
tai096	1508331	1490569	1506499	1,06	38810	38704	149,59	-1,18
tai097	1541419	1524555	1548666	1,56	38475	38459	149,94	-1,09
tai098	1533397	1507573	1529827	1,45	38569	38042	147,95	-1,68
tai099	1507422	1487669	1504042	1,09	39122	38697	148,37	-1,31
tai100	1520800	1509447	1511636	0,14	38110	35964	141,55	-0,75
Média				**0,96**			**143,64**	**-1,13**
200x20								
tai101	2012785	1997368	2010384	0,65	78700	78579	299,54	-0,77
tai102	2057409	2023201	2061397	1,85	77835	70313	271,01	-1,66
tai103	2050169	2024717	2034127	0,46	78075	70630	271,39	-1,24
tai104	2040946	2031515	2044056	0,61	77664	77020	297,51	-0,46
tai105	2027138	2020412	2031261	0,53	78554	64871	247,74	-0,33
tai106	2046542	2041153	2045448	0,21	78313	56412	216,10	-0,26
tai107	2045906	2025808	2043081	0,85	77732	77602	299,50	-0,98
tai108	2044218	2031402	2046303	0,73	77283	76874	298,41	-0,63
tai109	2037040	2010645	2034183	1,16	77866	77846	299,92	-1,30
tai110	2046966	2035128	2047852	0,62	77831	75553	291,22	-0,58
Média				**0,77**			**279,24**	**-0,82**

5.3. Experimento 3 – rAG x GASA e TS-M

Este experimento tem como objetivo analisar o desempenho do rAG no CPFSP, sendo o *makespan* o critério de desempenho, com os métodos de Grabowski e Pempera (2005), o TS-M descrito na Seção 3.3.6, e com o algoritmo híbrido desenvolvido por Shuster e Framinan (2003), GASA descrito na Seção 3.3.5. Os resultados do TS-M foram obtidos num computador Pentium 1.000 MHz. Os resultados do GASA foram obtidos num computador Athlon 1.400 MHz. Daí, os experimentos com o rAG utilizaram metade do tempo utilizado pelo TS-M para tornar a comparação justa.

Estes dois métodos foram escolhidos para serem comparados com o rAG devido aos seus testes terem sido realizados com as instâncias de Reeves (1995) e Heller (1960). A Tabela 5.12 mostra os resultados do rAG e a comparação com o GASA, enquanto a Tabela 5.13 mostra a comparação com o TS-M. Nas Tabelas 5.12 e 5.13 a coluna um mostra o nome das instâncias, a coluna dois mostra o número de tarefas e máquinas das instâncias, a coluna três mostra os resultados do GASA ou do TS-M, a coluna quatro mostra os tempos usados pelo GASA ou pelo TS-M, a coluna cinco mostra os resultados do rAG, a coluna seis mostra o tempo de execução usado pelo rAG, a coluna sete mostra a razão entre o tempo de execução do rAG e do GASA ou do TS-M multiplicado por dois, devido a consideração do computador utilizado nos experimentos do rAG ser duas vezes mais rápido do que os computadores utilizados pelos outros métodos, e a coluna oito mostra a diferença do desvio entre o rAG e o GASA ou o TS-M.

A análise da Tabela 5.12 produz as seguintes observações:

a) Nos 23 problemas testados o rAG obteve melhor resultado em 21 problemas e em 2 problemas obteve resultado igual ao GASA. Na média o rAG foi 4,99 % superior ao GASA;
b) O tempo de execução do rAG para os 23 problemas sempre foi inferior ao utilizado pelo GASA. A média da razão entre os tempos usados pelo rAG e pelo GASA foi 0,014 o que significa que o rAG é mais rápido do que o GASA;

c) A menor razão entre os tempos usados pelo rAG e o GASA foi de 0,003 o que significa que o rAG foi 333 vezes mais rápido que o GASA e, a maior razão foi 0,033 o que significa que o rAG foi 30 vezes mais rápido que o GASA;
d) O melhor resultado do rAG em comparação com o GASA foi -16,76% no problema hel1;
e) Quando o rAG foi melhor que o GASA a menor diferença ficou em -0,07%, no problema rec01; e
f) Os resultados do rAG tendem a serem melhores que o GASA quando o número de tarefas e máquinas é grande.

Estes resultados mostram claramente que o rAG é ao mesmo tempo mais eficaz e eficiente que o GASA, é mais eficaz porque obtém os melhores resultados e mais eficiente porque faz isso em menos tempo.

Tabela 5.12 – Comparação do rAG com o GASA.

Instância	n x m	GASA*	t_{GASA} (s)	rAG*	t_{rAG} (s)	$(t_{rAG}/t_{GASA}) \times 2$	Desvio (%)
rec01	20x5	1527	6,00	1526	0,10	0,033	-0,07
rec03	20x5	1392	6,00	1361	0,10	0,033	-2,23
rec05	20x5	1524	7,00	1514	0,10	0,029	-0,66
rec07	20x10	2046	12,00	2043	0,10	0,017	-0,15
rec09	20x10	2045	11,00	2042	0,10	0,018	-0,15
rec11	20x10	1881	10,00	1881	0,10	0,020	0,00
hel2	20x10	180	10,00	179	0,10	0,020	-0,56
rec13	20x15	2556	17,00	2545	0,15	0,018	-0,43
rec15	20x15	2529	17,00	2529	0,15	0,018	0,00
rec17	20x15	2590	16,00	2588	0,15	0,019	-0,08
rec19	30x10	2985	34,00	2850	0,20	0,012	-4,52
rec21	30x10	2948	35,00	2827	0,20	0,011	-4,10
rec23	30x10	2827	35,00	2703	0,20	0,011	-4,39
rec25	30x15	3732	55,00	3593	0,25	0,009	-3,72
rec27	30x15	3560	51,00	3431	0,25	0,010	-3,62
rec29	30x15	3440	54,00	3303	0,25	0,009	-3,98
rec31	50x10	4757	147,00	4343	0,55	0,007	-8,70
rec33	50x10	4998	145,00	4510	0,55	0,008	-9,76
rec35	50x10	4891	146,00	4420	0,55	0,008	-9,63
rec37	75x20	9508	907,00	8203	1,30	0,003	-13,73
rec39	75x20	9964	890,00	8554	1,30	0,003	-14,15
rec41	75x20	9978	904,00	8647	1,30	0,003	-13,34
hel1	100x10	877	1088,00	730	1,95	0,004	-16,76
Média			**200,13**		**0,43**	**0,014**	**-4,99**

Analisando os dados da Tabela 5.13 se descreve as seguintes observações:

a) Nos 23 problemas testados o rAG obteve melhor desempenho que o TS-M em 7 problemas, enquanto em 6 o desempenho foi idêntico e em 10 problemas o TS-M foi melhor. Na média o rAG foi 0,22% inferior ao desempenho do TS-M;
b) O tempo de execução do rAG para os 23 problemas foi comparativamente o mesmo usado pelo TS-M;
c) O melhor resultado do rAG, em comparação com o TS-M, foi -0,52% no problema rec19; e
d) O pior resultado do rAG em comparação com o TS-M foi 1,96%, no problema hel1.

Estes resultados mostram que o rAG é inferior ao TS-M quando as condições de tempo de execução são equivalentes. Porém a diferença é muito pequena de 0,22% e lembrando que o TS-M começa de uma solução boa.

Tabela 5.13 – Comparação do rAG com o TS-M.

Instância	n x m	TS-M*	t_{TS-M} (s)	rAG*	t_{rAG} (s)	(t_{rAG}/ t_{TS-M}) x 2	Desvio (%)
rec01	20x5	1527	0,20	1526	0,10	1,00	-0,07
rec03	20x5	1361	0,20	1361	0,10	1,00	0,00
rec05	20x5	1512	0,20	1514	0,10	1,00	0,13
rec07	20x10	2042	0,20	2043	0,10	1,00	0,05
rec09	20x10	2043	0,20	2042	0,10	1,00	-0,05
rec11	20x10	1881	0,20	1881	0,10	1,00	0,00
hel2	20x10	179	0,20	179	0,10	1,00	0,00
rec13	20x15	2545	0,30	2545	0,15	1,00	0,00
rec15	20x15	2529	0,30	2529	0,15	1,00	0,00
rec17	20x15	2587	0,30	2588	0,15	1,00	0,04
rec19	30x10	2865	0,40	2850	0,20	1,00	-0,52
rec21	30x10	2825	0,40	2827	0,20	1,00	0,07
rec23	30x10	2705	0,40	2703	0,20	1,00	-0,07
rec25	30x15	3593	0,50	3593	0,25	1,00	0,00
rec27	30x15	3432	0,50	3431	0,25	1,00	-0,03
rec29	30x15	3291	0,50	3303	0,25	1,00	0,36
rec31	50x10	4347	1,10	4343	0,55	1,00	-0,09
rec33	50x10	4469	1,10	4510	0,55	1,00	0,92
rec35	50x10	4427	1,10	4420	0,55	1,00	-0,16
rec37	75x20	8127	2,60	8203	1,30	1,00	0,94
rec39	75x20	8518	2,60	8554	1,30	1,00	0,42
rec41	75x20	8543	2,60	8647	1,30	1,00	1,22
hel1	100x10	716	3,90	730	1,95	1,00	1,96
Média			**0,87**		**0,43**	**1,00**	**0,22**

5.4. Experimento 4 – rAG x TS-M

O quarto tipo de experimento tem o objetivo de analisar a capacidade do rAG obter soluções melhores do que o TS-M. Por isso, foram utilizados tempos de execução maiores do que no experimento anterior. A Tabela 5.14 mostra os resultados do rAG e os compara com o TS-M. Na Tabela 5.14 a coluna um mostra o nome das instâncias, a coluna dois mostra o número de tarefas e máquinas das instâncias, a coluna três mostra os resultados do TS-M, a coluna quatro mostra os tempos usados pelo TS-M, a coluna cinco mostra os resultados do rAG, a coluna seis mostra o tempo de execução usado pelo rAG, a coluna sete mostra a razão entre o tempo de execução do rAG e do TS-M multiplicado por dois, porque se considera o computador utilizado no experimento do rAG duas vezes mais rápido do que o computador utilizado pelo TS-M, e a coluna oito mostra o desvio percentual entre o rAG e o TS-M.

Analisando as informações contidas na Tabela 5.14 tem-se que:

a) Nos 23 problemas testados o rAG obteve melhor resultado em 14 problemas, enquanto em 9 problemas obteve resultado igual ao TS-M. Na média o rAG foi 0,16% superior ao TS-M;
b) O tempo de execução do rAG para os 23 problemas sempre foi superior ao utilizado pelo TS-M. A média da razão entre os tempos usados pelo rAG e pelo TS-M foi 4,46 o que significa que o rAG foi mais lento que o TS-M;
c) A menor razão entre os tempos usados pelo rAG e o TS-M foi 2,00 e a maior razão foi 8,00;
d) O melhor resultado do rAG em comparação com o TS-M foi -0,71% no problema rec39; e
e) Quando o rAG foi melhor que o TS-M a menor diferença foi -0,05% no problema rec09.

Estes resultados mostram que o rAG é mais eficaz que o TS-M, porque obteve as melhores soluções quando comparadas com as soluções apresentadas pelo TS-M.

Tabela 5.14 – Comparação do rAG com o TS-M utilizando tempo maior de execução.

Instância	n x m	TS-M*	t_{TS-M} (s)	rAG*	t_{rAG} (s)	(t_{rAG}/ t_{TS-M}) x 2	Desvio (%)
rec01	20x5	1527	0,20	1526	0,20	2,00	-0,07
rec03	20x5	1361	0,20	1361	0,20	2,00	0,00
rec05	20x5	1512	0,20	1511	0,20	2,00	-0,07
rec07	20x10	2042	0,20	2042	0,20	2,00	0,00
rec09	20x10	2043	0,20	2042	0,20	2,00	-0,05
rec11	20x10	1881	0,20	1881	0,20	2,00	0,00
hel2	20x10	179	0,20	179	0,20	2,00	0,00
rec13	20x15	2545	0,30	2545	0,50	3,33	0,00
rec15	20x15	2529	0,30	2529	0,50	3,33	0,00
rec17	20x15	2587	0,30	2587	0,50	3,33	0,00
rec19	30x10	2865	0,40	2850	1,50	7,50	-0,52
rec21	30x10	2825	0,40	2821	1,50	7,50	-0,14
rec23	30x10	2705	0,40	2700	1,50	7,50	-0,18
rec25	30x15	3593	0,50	3593	2,00	8,00	0,00
rec27	30x15	3432	0,50	3431	2,00	8,00	-0,03
rec29	30x15	3291	0,50	3291	2,00	8,00	0,00
rec31	50x10	4347	1,10	4320	2,00	3,64	-0,62
rec33	50x10	4469	1,10	4458	2,00	3,64	-0,25
rec35	50x10	4427	1,10	4409	2,00	3,64	-0,41
rec37	75x20	8127	2,60	8069	7,00	5,38	-0,71
rec39	75x20	8518	2,60	8501	7,00	5,38	-0,20
rec41	75x20	8543	2,60	8514	7,00	5,38	-0,34
hel1	100x10	716	3,90	715	10,00	5,13	-0,14
Média			**0,87**		**2,19**	**4,46**	**-0,16**

5.5. Experimento 5 - rAG x outros AG

Neste tipo de experimento o objetivo é analisar o desempenho do rAG em relação a outros AGs quando aplicado no PFSP com o *makespan* sendo o critério de desempenho. Os resultados obtidos para a comparação foram retirados de Ruiz *et al.* (2006). Os experimentos com os outros AG foram realizados num computador Pentium IV (2,8 GHz e 512 MB de RAM). Os tempos de execução utilizados nos experimentos possuem três níveis representados pelos valores de *p*, são fornecidos em milissegundos e calculados por n*(m/2)*p, onde p = 30, 60 e 90. Considerou-se a velocidade do computador utilizado por Ruiz *et al.* (2006) como sendo a mesma do usado pelo rAG. Os problemas de teste utilizados foram às instâncias de Taillard (1993). Os AGs usados na comparação foram o GAChen de Chen *et al.* (1995), o GAMIT de Murata *et al.* (1996), o GAReev de Reeves (1995) e o GA_RMA de Ruiz *et al.* (2006), considerado o melhor AG sem hibridização encontrado na literatura. Estes AGs estão descritos na Seção 2.5 sendo que o quinto AG é o GA_AA de Aldowaisan e Allahverdi (2003), descrito na Seção 3.3.3 que foi desenvolvido para o CPFSP, mas que Ruiz *et al.* (2006) adaptaram para o PFSP.

A Tabela 5.15 mostra os tempos de execução para as 12 classes de problemas e os três valores de *p* utilizado. As Tabelas 5.16 a 5.18 mostram o resumo dos resultados do rAG e a comparação com os outros AGs. Nestas tabelas, a coluna um mostra a classe da instância, a coluna dois mostra o desvio do rAG em relação as melhores soluções encontradas nos problemas de Taillard (1993), as colunas seguintes mostram o desvio das soluções dos outros AGs e as diferenças dos seus desvios em relação ao desvio do rAG.

Tabela 5.15 – Tempos de execução utilizados nos testes com os outros AGs.

Instâncias	p = 30	p = 60	p = 90
	t (s)	t (s)	t (s)
20 x 5	1,50	3,00	4,50
20 x 10	3,00	6,00	9,00
20 x 20	6,00	12,00	18,00
50 x 5	3,75	7,50	11,25
50 x 10	7,50	15,00	22,50
50 x 20	15,00	30,00	45,00
100 x 5	7,50	15,00	22,50
100 x 10	15,00	30,00	45,00
100 x 20	30,00	60,00	90,00
200 x 10	30,00	60,00	90,00
200 x 20	60,00	120,00	180,00
500 x 20	150,00	300,00	450,00

Analisando as Tabelas 5.16 a 5.18 tem-se que:

a) No primeiro nível de tempo de execução (p= 30) o rAG foi superior ao GAChen, ao GAMIT e ao GA_AA e inferior ao GAReev e ao GA_RMA;
b) Nos segundo (p= 60) e terceiro (p= 90) níveis de tempo de execução o rAG só foi inferior ao GA_RMA;
c) O rAG em relação ao GAMIT só foi inferior na classe 100x5 em todos os níveis de tempo de execução;
d) O rAG em relação ao GA_AA só foi inferior na classe 100x5 em todos os níveis de tempo de execução e na classe 500x20 nos dois primeiros níveis de tempo de execução;
e) O rAG em relação ao GAReev foi superior nas classes 20x5, 20x10, 20x20 e 50x10 em todos os níveis de tempo de execução e na classe 50x5 no segundo nível de tempo de execução. Os resultados do rAG em relação ao GAReev são no segundo e terceiro níveis de tempo de execução tão melhores nas primeiras classes de problemas que fazem na média o rAG ser superior;
f) O rAG em relação ao GA_RMA foi superior nas classes 20x5, 20x10 e 20x20 em todos os níveis de tempo de execução e na classe 50x10 no segundo e terceiro níveis de tempo de execução. Diferente do GAReev estes resultados não são suficientes para fazer o rAG ser superior ao GA_RMA em qualquer um dos níveis de tempo de execução; e

g) Em comparação com os outros AGs o rAG tende a ser melhor quando o número de tarefas e máquinas é pequeno.

Estes resultados mostram que o rAG mesmo sem inicialização eficiente e hibridização consegue ser competitivo em relação aos outros AGs, sendo as vezes até melhor. Como já mencionado o GA_RMA é o melhor AG encontrado na literatura e mesmo assim o rAG em algumas instâncias conseguiu superá-lo. A média da diferença entre o rAG e o GA_RMA no terceiro nível de tempo de execução foi de 0,29%. Está diferença mostra que o rAG consegue ser competitivo em relação ao melhor AG encontrado na literatura.

Tabela 5.16 – Resultados dos experimentos com p = 30.

Instância (n x m)	rAG	GAChen	Difer rAG (%)	GAMIT	Difer rAG (%)	GA_AA	Difer rAG (%)	GAReev	Difer rAG (%)	GA_RMA	Difer rAG (%)
20 x 5	0,06	3,65	-3,59	0,84	-0,78	0,94	-0,88	0,54	-0,48	0,24	-0,18
20 x 10	0,33	5,00	-4,67	1,96	-1,63	1,70	-1,37	1,78	-1,45	0,62	-0,29
20 x 20	0,18	3,90	-3,72	1,66	-1,48	1,31	-1,13	1,39	-1,21	0,37	-0,19
50 x 5	0,22	1,89	-1,67	0,30	-0,08	0,37	-0,15	0,17	0,05	0,06	0,16
50 x 10	1,90	6,37	-4,47	3,50	-1,60	3,60	-1,70	2,23	-0,33	1,79	0,11
50 x 20	4,31	7,88	-3,57	5,07	-0,76	4,66	-0,35	3,74	0,57	2,67	1,64
100 x 5	0,35	1,34	-0,99	0,25	0,10	0,26	0,09	0,14	0,21	0,07	0,28
100 x 10	1,28	3,90	-2,62	1,54	-0,26	1,65	-0,37	0,82	0,46	0,65	0,63
100 x 20	4,30	8,06	-3,76	4,99	-0,69	4,92	-0,62	3,36	0,94	2,78	1,52
200 x 10	0,98	2,80	-1,82	1,14	-0,16	1,08	-0,10	0,59	0,39	0,43	0,55
200 x 20	3,87	6,94	-3,07	4,19	-0,32	3,95	-0,08	2,71	1,16	2,35	1,52
500 x 20	2,60	4,79	-2,19	2,68	-0,08	2,06	0,54	1,47	1,13	1,43	1,17
Média	**1,70**	**4,71**	**-3,01**	**2,34**	**-0,65**	**2,21**	**-0,51**	**1,58**	**0,12**	**1,12**	**0,58**

Tabela 5.17 – Resultados dos experimentos com p = 60.

Instância (n x m)	rAG	GAChen	Difer rAG (%)	GAMIT	Difer rAG (%)	GA_AA	Difer rAG (%)	GAReev	Difer rAG (%)	GA_RMA	Difer rAG (%)
20 x 5	0,04	4,02	-3,98	0,74	-0,70	0,80	-0,76	0,51	-0,47	0,23	-0,19
20 x 10	0,08	5,14	-5,06	1,72	-1,64	1,41	-1,33	1,67	-1,59	0,60	-0,52
20 x 20	0,08	3,93	-3,85	1,66	-1,58	1,37	-1,29	1,41	-1,33	0,34	-0,26
50 x 5	0,16	2,02	-1,86	0,26	-0,10	0,37	-0,21	0,20	-0,04	0,06	0,10
50 x 10	1,32	6,83	-5,51	3,20	-1,88	3,35	-2,03	2,26	-0,94	1,86	-0,54
50 x 20	4,11	7,98	-3,87	4,88	-0,77	4,52	-0,41	3,71	0,40	2,62	1,49
100 x 5	0,36	1,44	-1,08	0,25	0,11	0,24	0,12	0,12	0,24	0,08	0,28
100 x 10	0,89	3,78	-2,89	1,46	-0,57	1,61	-0,72	0,74	0,15	0,62	0,27
100 x 20	3,89	8,18	-4,29	4,77	-0,88	4,73	-0,84	3,25	0,64	2,68	1,21
200 x 10	0,88	2,75	-1,87	1,04	-0,16	1,10	-0,22	0,50	0,38	0,41	0,47
200 x 20	3,39	7,24	-3,85	4,14	-0,75	4,02	-0,63	2,65	0,74	2,22	1,17
500 x 20	2,19	4,79	-2,60	2,48	-0,29	1,98	0,21	1,38	0,81	1,40	0,79
Média	**1,45**	**4,84**	**-3,39**	**2,22**	**-0,77**	**2,13**	**-0,68**	**1,53**	**-0,08**	**1,09**	**0,36**

Tabela 5.18 – Resultados dos experimentos com p = 90.

Instância (n x m)	rAG	GAChen	Difer rAG (%)	GAMIT	Difer rAG (%)	GA_AA	Difer rAG (%)	GAReev	Difer rAG (%)	GA_RMA	Difer rAG (%)
20 x 5	0,10	3,51	-3,41	0,53	-0,43	0,84	-0,74	0,62	-0,52	0,25	-0,15
20 x 10	0,12	4,99	-4,87	1,61	-1,49	1,42	-1,30	1,71	-1,59	0,64	-0,52
20 x 20	0,04	4,24	-4,20	1,36	-1,32	1,23	-1,19	1,31	-1,27	0,40	-0,36
50 x 5	0,17	2,34	-2,17	0,23	-0,06	0,34	-0,17	0,16	0,01	0,06	0,11
50 x10	1,21	6,92	-5,71	3,27	-2,06	3,30	-2,09	2,00	-0,79	1,46	-0,25
50 x 20	3,67	7,77	-4,10	4,75	-1,08	4,69	-1,02	3,58	0,09	2,47	1,20
100 x 5	0,30	1,36	-1,06	0,22	0,08	0,22	0,08	0,11	0,19	0,06	0,24
100 x 10	0,84	3,87	-3,03	1,34	-0,50	1,55	-0,71	0,67	0,17	0,52	0,32
100 x 20	3,58	8,11	-4,53	4,68	-1,10	4,64	-1,06	3,12	0,46	2,54	1,04
200 x 10	0,72	2,81	-2,09	0,98	-0,26	0,99	-0,27	0,41	0,31	0,41	0,31
200 x 20	3,00	7,37	-4,37	3,95	-0,95	3,86	-0,86	2,54	0,46	2,11	0,89
500 x 20	2,00	4,62	-2,62	2,36	-0,36	2,08	-0,08	1,33	0,67	1,36	0,64
Média	**1,31**	**4,83**	**-3,51**	**2,11**	**-0,79**	**2,10**	**-0,78**	**1,46**	**-0,15**	**1,02**	**0,29**

5.6. Evolução das Soluções do rAG

Esta seção tem o objetivo de analisar a evolução das soluções obtidas pelo rAG nos problemas testados nos experimentos anteriores. Para isso é comparada a melhor solução obtida na população inicial com a solução final. A Tabela 5.19 mostra a média dos desvios entre a melhor solução obtida na população inicial e a solução final, do segundo tipo de experimento que foi realizado para o CPFSP, com o tempo total de fluxo como critério de desempenho. A Tabela 5.20 mostra a média dos desvios entre a melhor solução obtida na população inicial e a solução final, do quarto tipo de experimento que foi realizado para o CPFSP, como *makespan* sendo o critério de desempenho. A Tabela 5.21 mostra a média dos desvios entre a melhor solução obtida na população inicial e a solução final, do quinto tipo de experimento no terceiro nível de tempo de execução que foi realizado para o PFSP, com *makespan* como critério de desempenho.

Para mostrar a evolução das soluções foram construídos gráficos que mostram a melhoria do desvio em relação ao número de gerações. Foram escolhidas quatro instâncias de cada um dos três problemas avaliados. As Figuras 5.1 a 5.4 são referentes ao CPFSP, com o tempo total de fluxo como critério de desempenho. As Figuras 5.5 a 5.8 são referentes ao CPFSP, com o *makespan* como critério de desempenho. As Figuras 5.9 a 5.12 são referentes ao PFSP, com o *makespan* como critério de desempenho.

Desses experimentos foram observados que:

a) O rAG consegue melhorar bastante a qualidade da solução inicial. Deve-se levar em conta neste resultado que é mais fácil melhorar uma solução ruim que uma solução boa; e
b) Para os dois problemas CPFSP as melhores médias obtidas a partir da solução inicial foram 31,31% e 27,98%, respectivamente, maiores que a melhoria média obtida no PFSP que foi de 12,98%. Isto pode indicar uma característica diferente do espaço de soluções destes dois problemas.

As Figuras 5.1 a 5.12 mostram a evolução do desempenho do método a partir da solução inicial, mesmo quando o número de soluções já é bastante grande. Isto demonstra que o rAG consegue aproveitar todo o tempo de execução em prol da melhoria das soluções, ao invés de ficar em estado de estagnação.

Tabela 5.19 – Evolução das soluções do CPFSP com o tempo total de fluxo como critério de desempenho.

Instâncias (n x m)	D (rAG_{ini} e rAG*) (%)
20 x 5	-20,55
20 x 10	-19,43
20 x 20	-18,00
50 x 5	-31,95
50 x 10	-32,14
50 x 20	-30,68
100 x 5	-37,07
100 x 10	-36,95
100 x 20	-36,51
200 x 10	-40,45
200 x 20	-40,71
Média	**-31,31**

Tabela 5.20 – Evolução das soluções do CPFSP com o *makespan* como critério de desempenho.

Instâncias	n x m	D (rAG_{ini} e rAG*) (%)
rec01	20x5	-23,47
rec03	20x5	-28,33
rec05	20x5	-16,89
rec07	20x10	-24,03
rec09	20x10	-18,39
rec11	20x10	-23,35
hel2	20x10	-23,18
rec13	20x15	-26,38
rec15	20x15	-22,38
rec17	20x15	-21,39
rec19	30x10	-28,96
rec21	30x10	-27,91
rec23	30x10	-28,55
rec25	30x15	-29,09
rec27	30x15	-27,29
rec29	30x15	-32,89
rec31	50x10	-31,29
rec33	50x10	-32,82
rec35	50x10	-34,76
rec37	75x20	-35,32
rec39	75x20	-36,00
rec41	75x20	-35,08
hel1	100x10	-35,82
Média		**-27,98**

Tabela 5.21 – Evolução das soluções do PFSP com o *makespan* como critério de desempenho.

Instâncias (n x m)	D (rAG_{ini} e rAG*) (%)
20 x 5	-12,10
20 x 10	-14,79
20 x 20	-12,69
50 x 5	-8,74
50 x 10	-15,46
50 x 20	-15,81
100 x 5	-6,87
100 x 10	-12,71
100 x 20	-14,65
200 x 10	-9,92
200 x 20	-13,31
500 x 20	-10,34
Média	**-12,28**

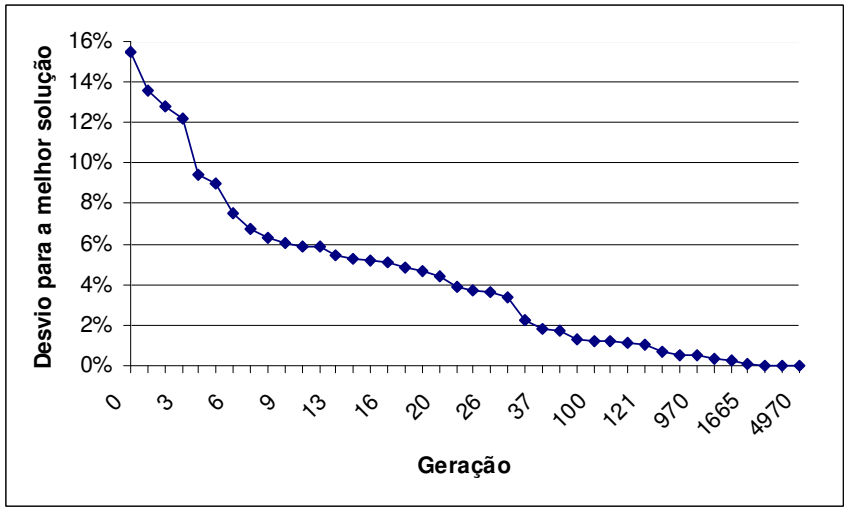


Figura 5.1 – Evolução das soluções do rAG para o problema tai021 (20x20).

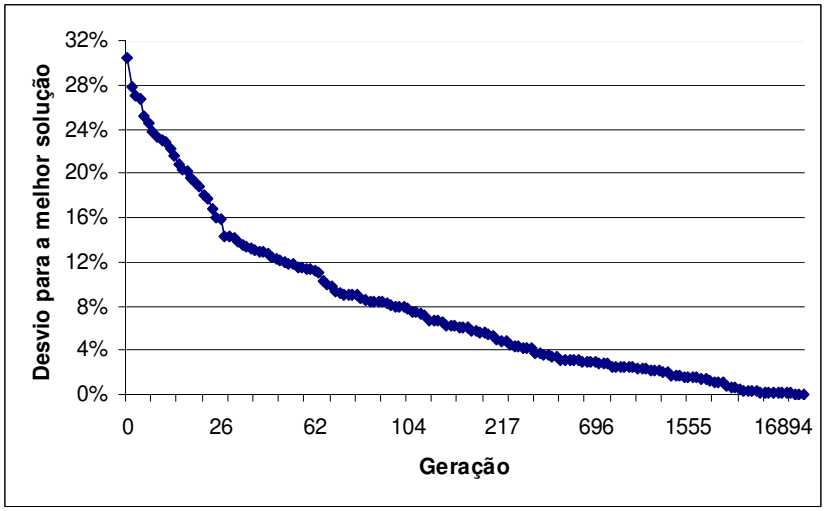


Figura 5.2 – Evolução das soluções do rAG para o problema tai051 (50x20).

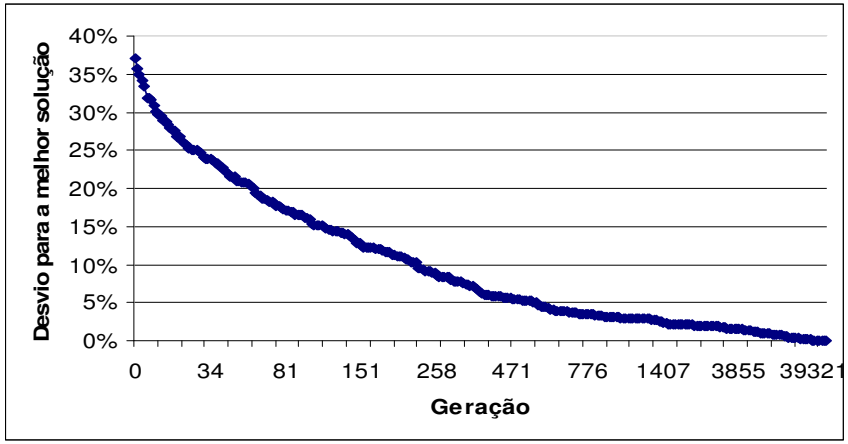


Figura 5.3 – Evolução das soluções do rAG para o problema tai081 (100x20).

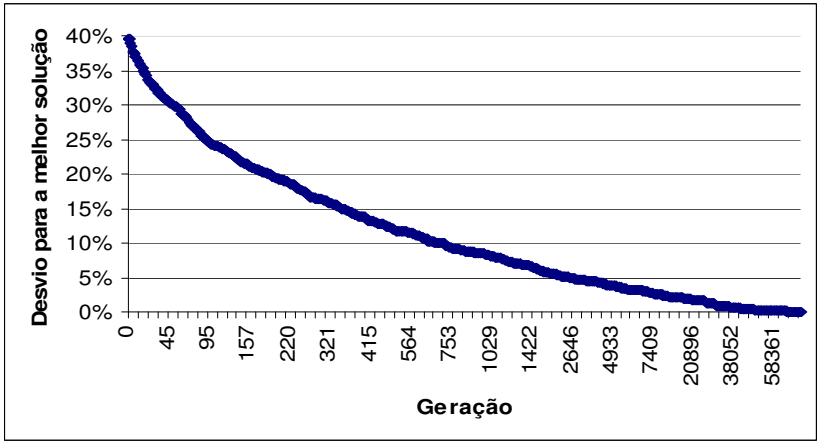


Figura 5.4 – Evolução das soluções do rAG para o problema tai101 (200x20).

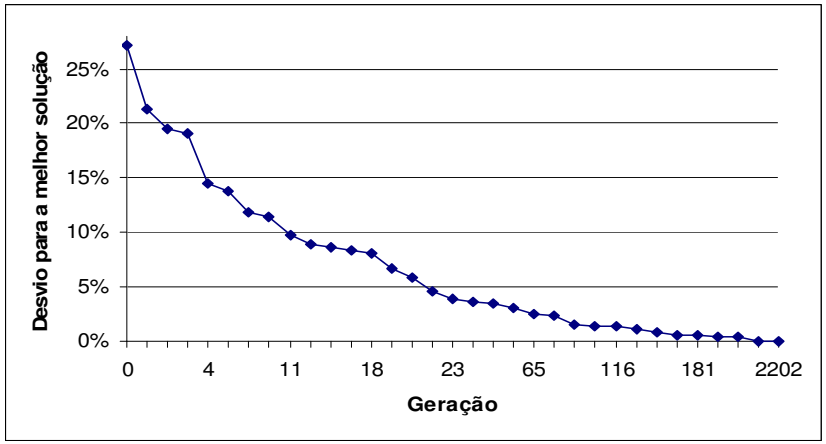


Figura 5.5 – Evolução das soluções do rAG para o problema rec17 (20x15).

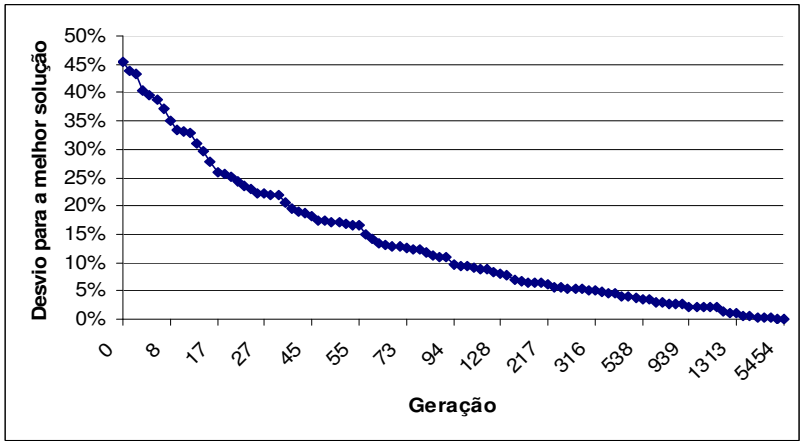


Figura 5.6 – Evolução das soluções do rAG para o problema rec31 (50x10).

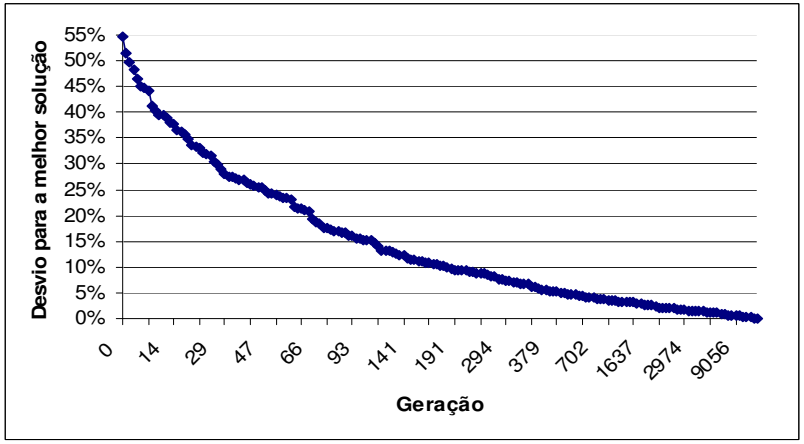


Figura 5.7 – Evolução das soluções do rAG para o problema rec37 (75x20).

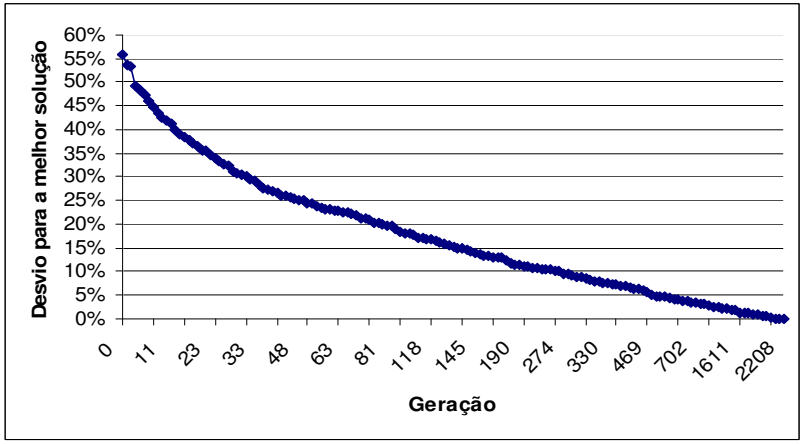


Figura 5.8 – Evolução das soluções do rAG para o problema hel1 (100x10).

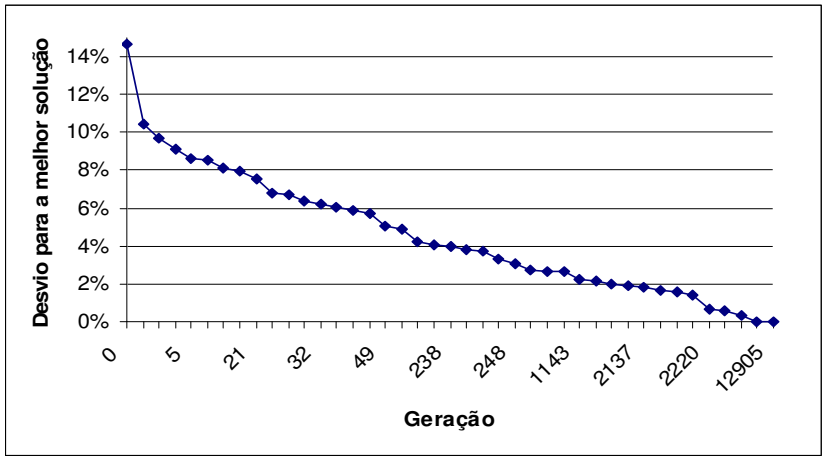


Figura 5.9 – Evolução das soluções do rAG para o problema tai021 (20x20).

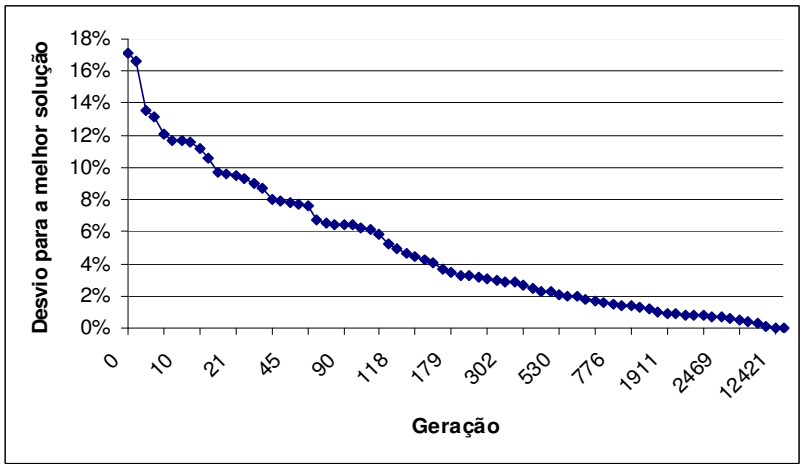


Figura 5.10 – Evolução das soluções do rAG para o problema tai051 (50x20).

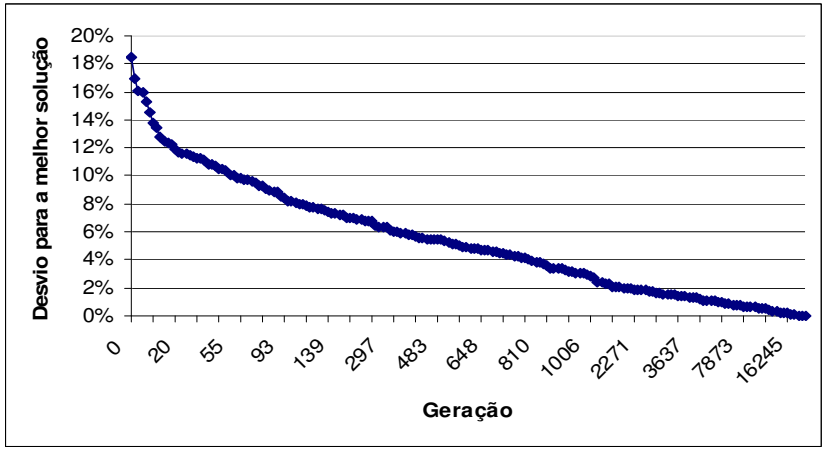


Figura 5.11 – Evolução das soluções do rAG para o problema tai081 (100x20).

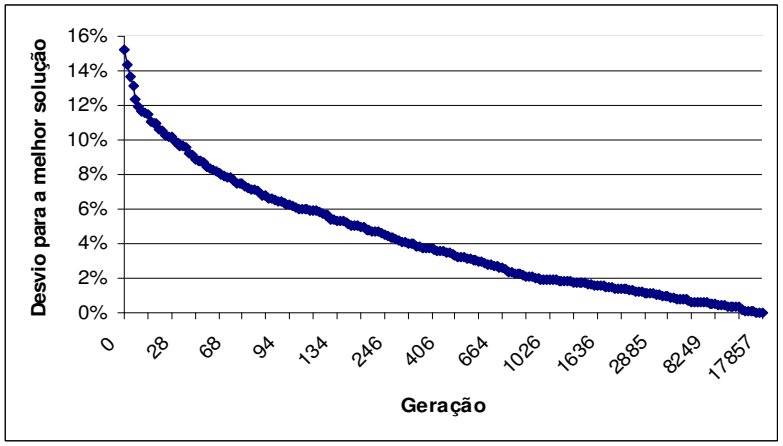


Figura 5.12 – Evolução das soluções do rAG para o problema tai101 (200x20).

5.7. Interpolação

Uma dificuldade de usar o rAG numa situação real é aquela em que o tempo de execução é um dado de entrada e a qualidade da solução é um dado de saída. Em termos práticos isso só é conhecido depois de decorrido o tempo de execução, quando a intenção seria obter a qualidade da solução com base no dado de entrada, sendo o tempo de execução. Para ajudar a enfrentar esta dificuldade foi construído um conjunto de funções para calcular o tempo de execução do rAG para o PFSP, a partir do número de tarefas, do número de máquinas e da qualidade desejada da solução. A interpolação polinomial foi à técnica usada para construir este conjunto de funções. A interpolação é usada quando não se conhece a expressão que define a função, mas só alguns valores da função que em geral são obtidos por experimentos previamente estabelecidos.

Segundo Cláudio e Marins (1994) o problema de interpolação pode ser definido da seguinte forma: fornecido um conjunto de dados (x_i, y_i), i = 1, 2, ...n, correspondentes aos valores de argumentos e valores de uma função *f*, tal que $y = f(x)$, deseja-se obter os valores f (x'), $x' \neq x_i$, utilizando os pontos dados. Assim, o objetivo da interpolação é obter o valor de f (x') aproximadamente. Para isso é construído, a partir do conjunto de dados, uma nova função *F* que interpola a função *f*, tal que:

$$\forall\ x_i,\ x_0 \leq x_i \leq x_n; \quad F(x_i) = f(x_i) \tag{5.1}$$

$$\forall\ x \in [x_0, x_n]; \qquad F(x_i) \cong f(x_i) \tag{5.2}$$

Para construir o conjunto de funções $F(x_i)$ foi escolhida a interpolação pelos polinômios de Lagrange. Dados n+1 pontos o polinômio de interpolação de Lagrange é dado pela Equação 5.3.

$$P_n = \sum_{i=0}^{n} a_i L_i(x), \tag{5.3}$$

onde:

$$L_k(x) = \prod_{i=0,\, k \neq i}^{n} \frac{x - x_i}{x_k - x_i} \tag{5.4}$$

A priori foi escolhido construir polinômios do segundo grau, onde são necessários três valores de x e f (x) para construir cada um dos polinômios. Estes valores são retirados dos resultados obtidos pelo rAG segundo os dados das Tabelas 5.16 a 5.18 onde são apresentados três valores obtidos para cada uma das 12 classes das instâncias de Taillard (1993). Na Tabela 5.22 a coluna um mostra o número de conjuntos de valores de x e f (x), a coluna dois mostra o número de tarefas, a coluna três mostra o número de máquinas, as colunas quatro a seis mostram os pontos x e f (x), onde x é o desvio e f (x) é o tempo de execução utilizado.

A Tabela 5.23 mostra os 12 polinômios construídos com os valores da Tabela 5.22. Na Tabela 5.23 a coluna um mostra o número do polinômio do segundo grau, a coluna dois mostra o intervalo de tarefas para o polinômio, a coluna três mostra o intervalo de máquinas para o polinômio, a coluna quatro mostra o intervalo de desvios para o polinômio e a coluna cinco mostra o polinômio do segundo grau.

A partir dos 12 polinômios do segundo grau mostrados na Tabela 5.23 é possível calcular o tempo de execução aproximado necessário para o rAG obter qualquer valor de qualidade de solução mostrado no intervalo. Podem-se calcular tempos de execução para problemas a partir de 1 tarefa e até 500 tarefas, processados em 1 máquina ou até 20 máquinas.

Para exemplificar a utilização dos polinômios, considere um problema com 18 máquinas e 25 tarefas para seqüenciar, o procedimento seria o seguinte:

i. Qual o intervalo de tarefas: $20 < n \leq 50$;
ii. Qual o intervalo de máquinas: $10 < m \leq 20$;
iii. Verifica-se na Tabela 5.22 qual é o polinômio que corresponde a esses dois intervalos: polinômio de número 6;
iv. Escolhe-se um desvio dentro do intervalo correspondente: por exemplo, 4%;
v. Aplica-se x= 4 no polinômio de número 6: $-63{,}920x(4)^2 + 463{,}210x(4) - 794{,}043 = 36{,}077$; e
vi. O Resultado é interpretado da seguinte forma: para um desvio de 4% são necessários 36,077 segundos de tempo de execução do rAG.

Tabela 5.22 – Valores usados para construir os polinômios do segundo grau.

Nº	n	m	P1 (x, f(x))	P2 (x, f(x))	P3 (x, f(x))
1	20	5	(0,06% ; 1,5s)	(0,04% ; 3,0s)	(0,10% ; 4,5s)
2	20	10	(0,33% ; 3,0s)	(0,08% ; 6,0s)	(0,12% ; 9,0s)
3	20	20	(0,18% ; 6,0s)	(0,08% ; 12,0s)	(0,04% ; 18,0s)
4	50	5	(0,22% ; 3,75s)	(0,16% ; 7,5s)	(0,17% ; 11,25s)
5	50	10	(1,90% ; 7,5s)	(1,32% ; 15,0s)	(1,21% ; 22,5s)
6	50	20	(4,31% ; 15,0s)	(4,11% ; 30,0s)	(3,67% ; 45,0s)
7	100	5	(0,35% ; 7,5s)	(0,36% ; 15,0s)	(0,30% ; 22,5s)
8	100	10	(1,28% ; 15,0s)	(0,89% ; 30,0s)	(0,84% ; 45,0s)
9	100	20	(4,30% ; 30,0s)	(3,89% ; 60,0s)	(3,58% ; 90,0s)
10	200	10	(0,98% ; 30,0s)	(0,88% ; 60,0s)	(0,72% ; 90,0s)
11	200	20	(3,87% ; 60,0s)	(3,39% ; 120,0s)	(3,00% ; 180,0s)
12	500	20	(2,60% ; 150,0s)	(2,19% ; 300,0s)	(2,00% ; 450,0s)

Tabela 5.23 – Polinômios do segundo grau para o problema PFSP.

Nº	n	m	x	Polinômio
1	$1 \leq n \leq 20$	$1 \leq m \leq 5$	(0,04%; 0,10%)	$2.500x^2 - 325x + 12$
2	$1 \leq n \leq 20$	$5 < m \leq 10$	(0,08%; 0,33%)	$-414,286x^2 + 157,857x - 3,977$
3	$1 \leq n \leq 20$	$10 < m \leq 20$	(0,04%; 0,18%)	$642,857x^2 - 227,143x + 26,057$
4	$20 < n \leq 50$	$1 \leq m \leq 5$	(0,16%; 0,22%)	$-8.750x^2 + 3.262,5x - 290,5$
5	$20 < n \leq 50$	$5 < m \leq 10$	(1,21%; 1,90%)	$80,074x^2 - 270,768x + 232,894$
6	$20 < n \leq 50$	$10 < m \leq 20$	(3,67%; 4,31%)	$-63,920x^2 + 463,210x - 794,043$
7	$50 < n \leq 100$	$1 \leq m \leq 5$	(0,30%; 0,36%)	$17.500x^2 - 11.675x + 1.950$
8	$50 < n \leq 100$	$5 < m \leq 10$	(0,84%; 1,28%)	$594,406x^2 - 1.328,322x + 741,378$
9	$50 < n \leq 100$	$10 < m \leq 20$	(3,58%; 4,30%)	$32,783x^2 - 341,660x + 892,988$
10	$100 < n \leq 200$	$1 \leq m \leq 10$	(0,72%; 0,98%)	$-432,692x^2 + 504,808x - 49,154$
11	$100 < n \leq 200$	$10 < m \leq 20$	(3,00%; 3,87%)	$33,156x^2 - 365,716x + 978,740$
12	$200 < n \leq 500$	$1 \leq m \leq 20$	(2,00%; 2,60%)	$706,033x^2 - 3.747,754x - 5.121,374$

5.8. Aplicação Prática

Silva (1996) desenvolveu um algoritmo para resolver o modelo matemático linear inteiro misto do PFSP. Silva (1996) aplicou este algoritmo num PFSP real de uma indústria têxtil do estado do Ceará. Os dados levantados por Silva (1996) foram da produção de fios do mês de novembro de 1991.

De acordo com Silva (1996) as principais informações do problema prático são:

a) Tipos de fios a serem produzidos: AP30/1M, AP40/1T, AP40/1M, AP57/1, LAP20/1, LAP24/1, LAP30/1, LAP40/1, LPP43/1 e LCY43/1; e
b) Tipos e quantidades de máquinas a serem utilizadas na produção dos fios: Carda de algodão (18), Carda de poliéster (1), Pré-passador (1), Penteadeira (36), Passador1 (12), Passador2 (12), Maçarroqueira (8), Filatório (58) e Conicaleira (8).

Dessas informações Silva (1996) definiu os dados de entrada do problema:

a) Tarefas: a quantidade em ton/mês de cada tipo de fio que tem que ser produzido;
b) Máquinas: a quantidade de tipos de máquinas disponíveis para a produção de fio e que no total são 9. A Figura 5.13 apresenta a descrição dos 9 tipos de máquinas; e
c) Matriz P_{ij}: calculada de acordo com a Equação 5.5 com seu valor dado em horas.

$$P_{ij} = (MN_j \times 8 \times 30) / TME_i, \ \forall\, i = 1, 2, ..., 9 \text{ e } j = 1, 2, ..., 10 \qquad 5.5$$

Onde:

MN_j: é a quantidade de máquinas necessária diariamente por turno para cada tipo de fio j; e

TME_i: é a quantidade de máquinas existentes de cada tipo i.

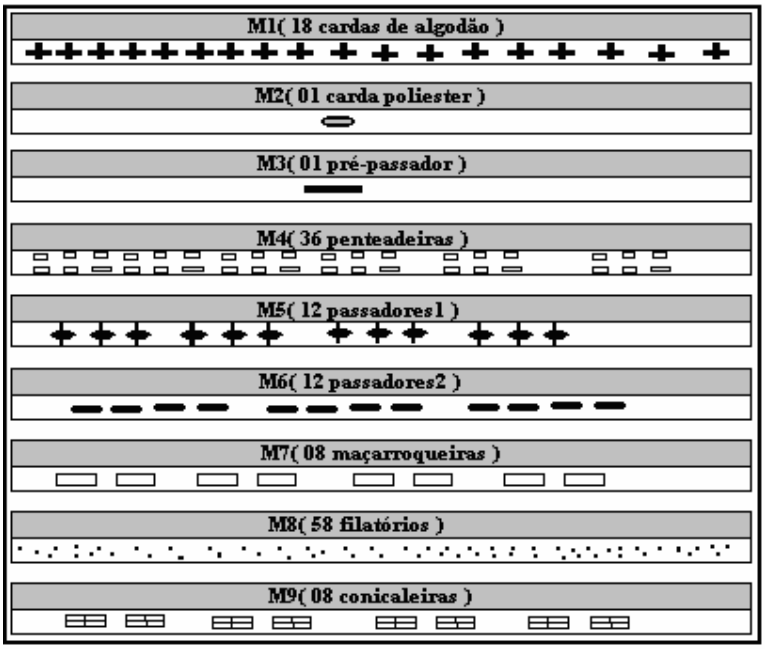


Figura 5.13 – Descrição das máquinas do problema prático. Fonte: Silva (1996).

A tabela 5.24 apresenta todos os valores de P_{ij} calculados a partir da Equação 5.6.

Tabela 5.24 – Tempos de execução em horas das tarefas do problema real. Fonte: Silva (1996).

Máq.	**Tarefas**									
	AP30/1 M	**AP40/ 1T**	**AP40/ 1M**	**AP57/ 1**	**LAP20/ 1**	**LAP 24/1**	**LAP 30/1**	**LAP40/ 1**	**LPP43/ 1**	**LCY43 /1**
M1	127.44	10.35	22.44	25.08	1.70	0.70	1.25	4.19	0.00	4.15
M2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	229.68	0.00
M3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	47.76	0.00
M4	53.12	4.31	9.35	9.95	0.72	0.29	0.54	1.80	0.00	1.69
M5	34.24	2.78	6.02	10.36	0.70	0.28	0.52	1.72	3.96	1.62
M6	36.14	2.94	6.36	10.30	0.68	0.28	0.52	1.72	3.96	1.62
M7	111.15	13.53	29.34	42.24	1.20	0.48	2.10	7.05	9.18	6.63
M8	95.43	15.50	29.90	53.88	0.87	0.41	1.10	7.40	15.39	16.19
M9	112.86	10.86	25.47	45.87	1.32	0.60	1.29	5.07	14.19	17.04

O problema era formado por 10 tarefas e 9 máquinas. O número de soluções possíveis para este problema (n= 10) é 10! = 3.628.800. Silva (1996) encontrou a solução ótima que é igual a 654 horas.

Usou-se um dos polinômios do segundo grau, construídos na Seção 5.7 para calcular o tempo de execução do rAG para resolver o problema apresentado em Silva (1996). Foi usado o segundo polinômio da Tabela 5.23 dado que n=10 e m=9. O polinômio é definido no intervalo (0,08%; 0,33%). Foi escolhido para o valor do desvio 0,10%, i.e., x=0,10. Com isso o tempo de execução calculado foi de 11,39 segundos, mostrado na Equação 5.6.

$$F(0{,}10\%) = -414{,}286 \times (0{,}10)^2 + 157{,}857 \times (0{,}10) - 3{,}977 = 11{,}39 \qquad 5.6$$

A Tabela 5.25 mostra os resultados obtidos pelo rAG nas cinco execuções realizadas. Nesta tabela mostra que o rAG obteve a solução ótima do problema que é 653,95h em todas as cinco execuções, a diferença em relação a solução de Silva (1996) deve-se ao arredondamento. A seqüência de tarefas obtidas na primeira execução foi 7 8 10 4 3 2 1 6 9 5 e está ilustrada na Figura 5.14. A melhor solução da população inicial obtida pelo rAG na primeira execução foi 654,05h e já na primeira geração o rAG obteve a solução ótima de 653,95 horas, ou seja, uma melhoria de 0,015%.

Tabela 5.25 – Resultados obtidos pelo rAG para o problema real.

Nº da execução	Nº de gerações	Tempo (s)	Solução
1	41.541	11,391	653,95
2	41.578	11,390	653,95
3	41.547	11,391	653,95
4	41.430	11,391	653,95
5	41.572	11,390	653,95

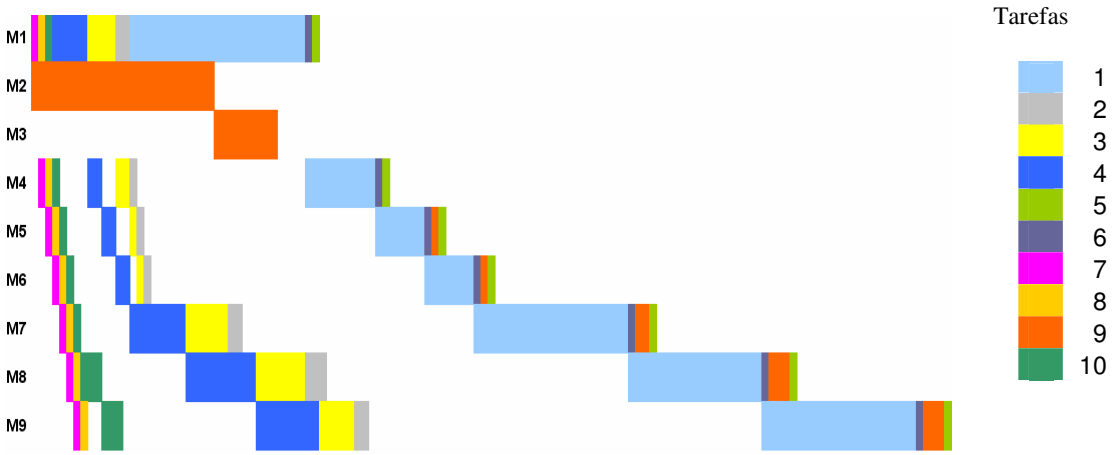


Figura 5.14 – Gráfico de Gantt para a solução do problema prático encontrada pelo rAG.

5.9. Conclusão dos Experimentos Computacionais

A comparação dos resultados do rAG com os resultados da heurística Pilot-10-Chins de Fink e Voβ (2003) para o CPFSP com o tempo total de fluxo como critério de desempenho demonstrou que o rAG foi 0,56% melhor, caso raro quando as condições de tempo de execução são equivalentes, porque normalmente as heurísticas são mais rápidas que o AG porque usam o conhecimento do problema para construir as suas soluções, enquanto o AG trabalha apenas sobre as estruturas da população, sem nenhuma hipótese definida *a priori* sobre o problema. Neste caso não seria vantajoso usar a heurística Pilot-10-Chins para gerar a população inicial do rAG. Este resultado também serve como uma sugestão prática: antes de usar uma heurística para gerar a população inicial de um AG, comparar o desempenho do AG com a população inicial gerada aleatoriamente e com a população gerada pela heurística escolhida. Usar a população inicial aleatória neste trabalho foi uma forma de melhorar a eficiência do AG sem depender de uma boa heurística para gerar a população inicial.

O rAG foi 0,34% melhor que o *Tabu Search* com solução inicial obtida pela heurística Pilot-10-Chins que é o melhor método de Fink e Voβ (2003) para o CPFSP como tempo total de fluxo como critério de desempenho. O rAG usou 20% do tempo de execução que Fink e Voβ (2003) usaram. Este resultado reforça a qualidade do rAG porque a comparação foi com um método que além de usar uma solução inicial boa só trabalha com uma solução de cada vez e, por isso, normalmente deveria ser mais rápido que um AG.

A única comparação possível entre o rAG e outro AG para o CPFSP é com o GASA, desenvolvido por Shuster e Framinan (2003). O rAG foi 4,99% melhor que o GASA, utilizando apenas 1,4% do tempo de execução utilizado pelo GASA. Este resultado mostra a diferença de qualidade entre os projetos dos dois AGs, sem esquecer que o GASA usa uma etapa de hibridização com a metaheurística *Simulated Annealing*, isto mostra que quando o projeto do AG não consegue aproveitar as suas qualidades teóricas uma etapa de hibridização não torna o desempenho do AG satisfatório.

No primeiro experimento do rAG comparado ao *Tabu Search* com multimovimento (TS-M) de Grabowski e Pempera (2005) para o CPFSP com o *makespan* como critério de desempenho, o rAG foi 0,22% inferior. Mesmo sendo uma diferença pequena este resultado mostra que realmente é difícil para um AG superar um método de busca em vizinhança. Mas este resultado também serve para mostrar que o AG pode ficar muito próximo a estes métodos, o que o resultado do GASA não mostrava.

Foram realizados mais experimentos com o rAG para o CPFSP com o *makespan* como critério de desempenho e dessa vez foram utilizados tempos de execução maiores. Nesta condição o rAG foi 0,16% melhor que o TS-M, utilizando 4,46 vezes mais tempo de execução. Este resultado mostra que o rAG consegue obter soluções cada vez melhores quando mais tempo de execução é utilizado, ao invés da busca tornar-se ineficaz por causa da convergência prematura.

Como o rAG obteve bons resultados para o CPFSP decidiu-se compará-lo com outros bons AG para saber qual a sua situação em relação a eles. Existem muitos bons AGs para problemas permutacionais aplicados ao PFSP, por isso, foi necessário fazer experimentos com este problema. Isto demonstra a característica generalista do AG, porque com uma pequena modificação foi possível testar o rAG em outro problema. Os experimentos foram realizados em três níveis de tempo de execução.

O rAG foi melhor que o GAChen de Chen *et al.* (1995) em todos os três níveis de tempo de execução. O melhor desempenho comparativo do rAG em relação a outro AG foi de -3,51% com o GAChen no terceiro nível de tempo de execução, a suposição para esta diferença é a utilização pelo GAChen das heurísticas CDS (Campbell *et al.*, 1970) e

Dannembing (Dannembring, 1977) para gerar a população inicial, que encontram soluções inferiores a heurística NEH de Nawaz *et al.* (1983) que é usada por outros dois AGs e não usar hibridização que é utilizada por outros dois AGs. Este resultado mostra a dependência que um AG pode ter da heurística que gera a população inicial ou de uma etapa de hibridização e que o rAG consegue ser eficiente sem usar dessas duas estratégias.

O rAG foi melhor que o GAMIT de Murata *et al.* (1996) nos três níveis de tempo de execução. A maior diferença entre o rAG e o GAMIT foi de 0,79%. O GAMIT utiliza população inicial gerada aleatoriamente, mas em compensação utiliza uma etapa de hibridização com busca local para aumentar a qualidade da solução final, mesmo assim, o rAG conseguiu obter soluções melhores que o GAMIT. Este resultado mostra que o rAG conseguiu ser mais eficiente mesmo sem utilizar uma etapa de hibridização.

O rAG foi melhor que o GA_AA de Aldowaisan e Allahverdi (2003) nos três níveis de tempo de execução. A maior diferença entre o rAG e o GA_AA foi de 0,78%. O GA_AA utilizou inicialização eficiente e uma etapa de hibridização com busca local. Este resultado mostra que o rAG conseguiu ser mais eficiente mesmo sem usar inicialização eficiente e uma etapa de hibridização.

O rAG foi melhor que o GAReev de Reeves (1995) em dois níveis de tempo de execução. No primeiro nível de tempo de execução o GAReev foi 0,12 melhor que o rAG. No segundo nível de tempo de execução o rAG foi 0,08% melhor que o GAReev e no terceiro nível foi 0,15% melhor. Não se pode afirmar com certeza, mas talvez o fato do GAReev ser o primeiro AG a ser melhor que o rAG na comparação com o PFSP seja porque é o primeiro AG nessa comparação a usar a heurística NEH de Nawaz *et al.* (1983) para gerar a população inicial. Este resultado comprova a teoria que a inicialização eficiente acelera a obtenção de boas soluções, mas pode comprometer a qualidade da solução final quando a eficiência do algoritmo é especialmente dependente da solução inicial. O rAG conseguiu melhorar a qualidade da solução obtida a medida que o tempo de execução aumentava o que o GAReev não conseguiu fazer na mesma intensidade.

O rAG foi inferior ao GA_RMA de Ruiz *et al.* (2006) nos três níveis de tempo de execução. A menor diferença entre o rAG e o GA_RMA foi de 0,29% no terceiro nível de

tempo de execução e a maior diferença foi de 0,58% no primeiro nível de tempo de execução. O GA_RMA é o segundo AG nesta comparação a usar a heurística NEH de Nawaz *et al.* (1983) para gerar a população inicial. A comparação do rAG com o GA_RMA comprova que esse é melhor que o rAG para o PFSP. Mesmo assim o rAG foi melhor que o GA_RMA nas classes 20x5, 20x10 e 20x20 em todos os níveis de tempo de execução e na classe 50x10 nos segundo e terceiro níveis de tempo de execução. Este resultado significa que para os problemas com até 20 tarefas e 20 máquinas o rAG é melhor que o GA_RMA para o PFSP.

CAPÍTULO 6 – CONCLUSÕES

O rAG mostrou que um AG que utiliza os seus princípios originais, diversificação e intensificação, de forma eficiente consegue obter bons resultados. Isto ficou comprovado com a implementação dos três procedimentos inspirados nesses princípios, que no primeiro experimento diminuiu o desvio das soluções do rAG de 1,710% para 0,171%, uma melhoria de 10 vezes. Este resultado aponta a importância de um bom projeto para os componentes originais do AG, antes de recorrer a inicialização eficiente e hibridização para tornar o AG competitivo em relação a outros métodos de otimização.

O rAG mostrou também que é possível um AG ser mais eficiente que uma heurística para o CPFSP sendo o tempo total de fluxo o critério de desempenho, pois foi 0,56% melhor que a heurística Pilot-10-Chins usando apenas 44% do tempo de execução utilizado por essa heurística. Mostrando que pelo menos para esse problema a opinião de alguns autores (Reeves e Rowe, 2002; Dréo *et al.*, 2006) que afirmam que um AG não consegue ser melhor que uma heurística nas mesmas condições de tempo de execução está equivocada.

O rAG mostrou ser o melhor método para o CPFSP sendo o tempo total de fluxo o critério de desempenho, porque seus resultados na média foram 0,34% superiores ao melhor método encontrado na literatura para esse problema que é o *Tabu Search* de Fink e Voβ (2003).

O rAG mostrou ser o melhor AG para o CPFSP sendo o *makespan* o critério de desempenho, porque foi 4,99% melhor que o AG de Shuster e Framinan (2003). Quando se trata da comparação com o melhor método para esse problema, o rAG, nas mesmas condições de tempo de execução foi 0,22% inferior ao TS-M de Grabowski e Pempera (2005). Mas o rAG se torna o melhor método para esse problema quando usa 4,46 vezes mais tempo de execução, porque apresenta um desvio médio 0,16% melhor que o TS-M.

O rAG em comparação aos outros cinco AG para o PFSP mostrou-se ser bem competitivo, sendo superado apenas pelo GA_RMA de Ruiz *et al.* (2006). A menor diferença média dos desvios entre o GA_RMA e o rAG foi de 0,29%. Vale salientar que o GA_RMA tem inicialização eficiente e um processo chamado de *restart* que realiza uma busca local.

A análise da evolução das soluções do rAG desde a solução inicial até a solução final mostrou que ele tem a capacidade de melhorar bastante a qualidade das soluções, mesmo depois que a solução já ter atingido uma boa qualidade. Esta análise também mostrou que para os problemas avaliados, rapidamente o rAG obtém boas soluções, motivo pelo qual o rAG ser mais eficiente que a heurística Pilot-10-Chins para o CPFSP sendo o tempo total de fluxo o critério de desempenho.

Por tudo isso o objetivo de construir um AG eficiente para os dois problemas sem população inicial gerada por uma boa heurística e nem hibridização foi cumprido. Este sucesso é atribuído aos três procedimentos propostos que foram capazes de manter a diversidade na população e ao mesmo tempo intensificar o processo de busca.

Além disso, apresentamos 12 polinômios de grau 2 que ajudam no processo de calcular o tempo de execução necessário para o rAG obter uma solução de determinada qualidade para o PFSP, a partir do número de tarefas, do número de máquinas e do desvio da solução dentro de um intervalo pré-definido. Estes polinômios podem ser usados para apoiar a decisão de quanto tempo de execução utilizar para o rAG em problemas reais.

O rAG foi testado num problema PFSP real de uma indústria têxtil cearense. O tempo de execução foi calculado a partir de um dos polinômios de grau 2 construído. O resultado do rAG no problema real foi muito satisfatório, pois na primeira geração já foi encontrada a solução ótima. Este resultado mostrou a qualidade do rAG, pois entre mais de 3 milhões de soluções possíveis rapidamente encontrou aquela que era a ótima.

Outro resultado da aplicação do rAG no problema prático foi mostrar que compensa desenvolver um algoritmo testando-o em problemas teóricos, pois quando é aplicado em um problema prático se mostra bastante eficiente.

Propostas para futuros trabalhos:

- Incorporar o atributo reativo à mutação populacional. Pois a mutação realiza apenas uma perturbação e sempre no mesmo intervalo de gerações sem melhoria. Uma sugestão é poder realizar mais de uma perturbação de cada vez e em intervalos de geração diferentes, dependendo da quantidade de tarefas e máquinas do problema;
- Como o rAG obteve bons resultados para o PFSP mesmo sem inicialização eficiente, uma proposta para melhorar o rAG seria implementar uma inicialização eficiente que não comprometa a diversidade da população;
- Poderia ser testados outros valores para os parâmetros do rAG;
- Aplicar o rAG em outros problemas reais de maior tamanho para verificar o seu desempenho; e
- Aplicar o rAG em outros problemas POCP e comparar o seu resultado com os melhores métodos desses problemas.

REFERÊNCIAS BIBLIOGRÁFICAS

AARTS, E.; LENSTRA, J.K. **Local search in combinatorial optimization**. Wiley Interscience, Chichester, England. 1997.

ALDOWAISAN, T; ALLHVERDI, A. (2003). New heuristics for no-wait flowshops to minimize makespan. **Computers and Operations Research**, v.30, p.1219-1231.

ALDOWAISAN, T; ALLHVERDI, A. (2004). New heuristics for m-machine no-wait flowshop to minimize total completion time. **The International Journal of Management Science - Omega**, v.32, p.345-352.

BATTITI, R. (1996). **Reactive search: toward self-tunning heuristics**. In: RAYWARD-SMITH, V.; OSMAN, I.; REEVES, C.; SMITH, G.. Modern Heuristic Search Methods. Wiley, Chichester, p.61-83.

BRUCKER, P. **Scheduling Algorithms**. Springer-Verlag, Berlin. 1998.

CAMPBELL, H.G.; DUDEK, R.A.; SMITH, M.L. (1970). An heuristic algorithm for n job m machine sequencing problem. **Management Science**, v.16, p.630-637.

CHEN, C.-L.; VEMPATI, V.S.; ALJABER, N. (1995). An application of genetic algorithms for flow shop problems. **European Journal of Operational Research**, v.80, p.389-396.

CHEN, C.-L.; NEPPALLI, R.V; ALJABER, N. (1996). Genetic algorithms applied to the continuous flow shop problem. **Computers and Industrial Engineering**, v.30, p.919-929.

CHAKRAVARTHY, K.; RAJENDRAN, C. (1999). A heuristic for scheduling in a flowshop with the bicriteria of makespan and maximum tardiness minimization. **Production Planning and Control**, v.10, p.707-714.

CLÁUDIO, D.M.; MARINS, J.M. **Cálculo numérico computacional : teoria e prática**. Atlas, São Paulo, 464 pp., 1994.

CLEVELAND, G.A.; SMITH, F. (1989). Using genetic algorithms to schedule flow shop release. Proc. 3rd Int. Conf. on Genetic Algorithms Aplications, 160-169.

CONWAY, R.W.; MAXWELL, W.L.; MILLER, L.W. **Theory of scheduling**. Reading, MA: Addison-Wesley; 1967.

DANNENBRING, D.G. (1977). An evaluation of flow shop sequencing heuristics. **Management Science**, v.23, p.1174-1182.

DAVIS, L. **Handbook of genetic algorithms**. Van Nostrand Reinhold, New York, 1991.

DEJONG, K.A. (1980). Adaptive systems design: a genetic approach. **IEEE Trans. Syst., Man, Cyber**. SMC-16, p.566-574.

DRÉO, J.; PÉTROWSKI, A.; SIARRY, P.; TAILLARD, E. **Metaheuristics for Hard Optimization: Methods and Case Studies**. Springer, New York, 369 pp., 2006.

DUDEK, R.A.; TEUTON Jr., O.F. (1964). Development of m-stage decision rule for scheduling n jobs through m machines. **Operations Research**, v.12, p.471-497.

DUIN, C.W.; VOβ, S. (1999). The pilot method: a strategy for heuristic repetition with application to the Steiner problem in graphs. **Networks**, v.34, p.181-191.

EVOWEB. Success breeds success. Disponível em : <http://evonet.lri.fr/evoweb/news_events/news_features/article.php?id=59>. Acesso em 4 de junho de 2007.

FINK, A.; VOβ, S. (2002). **HotFrame: A Heuristic Optimization Framework**. In: Voβ, S., Woodruff, D. Optimization Software Class Libraries. Kluwer, Boston, p.81-154.

FINK, A.; VOβ, S. (2003). Solving the continuous flow-shop scheduling problem by metaheuristics. **European Journal of Operational Research**, v.151, p.400-414.

GANGADHARAN, R.; RAJENDRAN, C. (1993). Heuristic algorithms for scheduling in the no-wait flowshop. **International Journal of Production Economics**, v.32, p.285-290.

GAREY, M.R.; JOHNSON, D.S. **Computers and Intractability: a Guide to the Theory of NP-completeness**. W.H. Freeman and Company, San Francisco, CA, 338 pp., 1979.

GLOVER, F.; LAGUNA, M. **Tabu Search**. Kluwer Academic Publishers, Dordrecht, 382 pp., 1997.

GOLDBERG, D.E. **Genetic Algorithms in Search, Optimization, and Machine Learning**. Addison Wesley, Reading, MA, 372 pp., 1989.

GRABOWSKI, J.; PEMPERA, J. (2000). Sequencing of jobs in some production system. **European Journal of Operational Research**, v.125, p.535-550.

GRABOWSKI, J.; PEMPERA, J. (2005). Some local search algorithms for no-wait flow-shop problem with makespan criterion. **Computers and Operations Research**, v.32, p.2197-2212.

GRABOWSKI, J.; WODECKI, M. (2004). A very fast tabu search algorithm for the flow shop problem with makespan criterion. **Computers and Operations Research**, v.11, p.1891-1909.

GRAHAM, R.L.; LAWLER, E.L.; LENSTRA, J.K.; KAN, A.H.G.R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics, 5:287–326.

GREFENSTETTE, J.J. (1986). Optimized control of parameters for genetic algorithms. **IEEE Trans. Syst., Man, Cyber**. SMC-16, p.122-128.

GUPTA, J.N.D. (1979). **A review of flowshop scheduling research**. In: RITZMAN, L.P.; KRAJEWSKI, L.J.; BERRY, W.L.; GOODMAN, S.T.; HARDY, S.T.; VITT, L.D. (Eds.) Disaggregation Problems in Manufacturing and Service Organizations. Martinus Nijhoff, The Hague, pp. 363–388.

GUPTA, J.N.D.; STAFFORD Jr, E.F. (2006). Flowshop scheduling research after five decades. **European Journal of Operational Research**, v.169, p.699–711.

HALL, N.G.; SRISKANDARAJAH, C. (1994). A survey of machine scheduling problems with blocking and no-wait in process. **Operations Research**, v.44, p.510-525.

HAUPT, R.L; HAUPT, S.E. **Pratical genetic algorithms**. John Wiley & Sons Inc., Hoboken, New Jersey, USA, 2ª ed., 253 pp., 2004.

HELLER, J. (1960). Some experiments for an M x J flow shop and its decision-theoretical aspects. **Operations Research**, v.8, p.178-184.

HOLLAND, J.H. **Adaptation in natural artificial systems**. University of Michigan Press, Michigan, 211 pp., 1975.

JOHNSON, D.S.; ARAGON, C.R.; McGREOCH, L.A.; SCHEVON, C. (1989). Optimization by simulated annealing: an experimental evaluation; part 1, graph partitioning. **Operations Research**, v.37, p.865-892.

JOHNSON, S.M. (1954). Optimal two- and three-stage production schedules with setup times included. **Naval Research logistics Quarterly** 1, p.61–68.

KIRKPATRICK, S.; GELATT Jr., C; VECCHI, M. (1983). Optimization by simulated annealing. **Science**, v.220, p.671-680.

LAWLER, E.L.; LENSTRAS, J.K.; RINNOOY KAN, A.H.G.; SHMOYS, D.B., 1993. **Sequencing and scheduling: Algorithms and complexity**. In: Graves, S.C. (Ed.), Handbooks in Operations Research and Management Science, Vol. 4. Elsevier Science Publishers, Amsterdam, pp. 445–552.

LENSTRA, J.K.; RINNOOY KAN, A.H.G.; BRUCKER, P. (1977). Complexity of machine scheduling problems. Annals of Discrete Mathematics 1, p.343-362.

LOBO, F.G.; GOLDBERG, D.E. (2004). The parameter-less genetic algorithm in practice. **Information Sciences**, v.167, p.217-232.

LOMNICKI, Z.A. (1965). A branch and bound algorithm for the exact solution of the three machine scheduling problem. **Operational Research Quarterly**, v.16, p.89-100.

MANNE, A. (1960). On the job-shop scheduling problem. **Operations Research**, v.8, p.219–223.

MITCHELL, M. **An introduction to genetic algorithms**. MIT Press, Cambridge, Massachusetts, USA, 158 pp., 1998.

MURATA, T.; ISHIBUCHI, H.; TANAKA, H. (1996). Genetic algorithms for flowshop scheduling problems. **Computers & Industrial Engineering**, v.30, p.1061-1071.

MUTH, J.; THOMPSON, G.L. **Industrial Scheduling**. Prentice-Hall, Englewood Cliffs, New Jersey, 1963.

NAWAZ, M.; ENSCORE, E.; HAM, I. (1983). A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. **OMEGA. The International Journal of Management Sciences**, v.11, p.91-95.

PANWALKAR, S.S.; WOOLLAM, C.R. (1980). Ordered flow shop problem with no in-process waiting: further results. **Journal of the Operational Research Society**, v.31, p.1039-1043.

PAPADIMITRIOU, C.H.; KANELLAKIS, P.C. (1980). Flowshop scheduling with limited temporary storage. **Journal of the Association for Computing Machinery**, v.27, p.533-549.

PICARD, J.-C; QUEYRANNE, M. (1978). The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. **Operations Research**, v.26, p.86-110.

RAJENDRAN, C. (1994). A no-wait flowshop scheduling heuristic to minimize makespan. **Journal of the Operational Research Society**, v.45, p.472-478.

RAJENDRAN, C.; CHAUDHURI, D. (1990). Heuristic algorithms for continuous flow-shop problem. **Naval Research Logistics**, v.37, p.695-705.

RAJENDRAN, C.; ZIEGLER, H. (1997). An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. **European Journal of Operational Research**, v.103, p.129-138.

REEVES, C.R. (1995). A genetic algorithm for flow shop sequencing. **Computers and Operations Research**, v.22, p.5-13.

REEVES, C.R.; ROWE, J.E. **Genetic Algorithms: Principles and Perspectives : a Guide to GA Theory**. Kluwer Academic Publishers, New York, 332 pp., 2002.

RÖCK, H. (1984). The three-machine no-wait flowshop problem is NP-complete. **Journal of the Association for Computing Machinery**, v.31, p.336-345.

ROTHLAUF, F. **Representations for genetic and evolutionary algorithms**. Spring-Verlag, Berlim, 325 pp., 2006.

RUIZ, R.; MAROTO, C.; ALCARAZ, J. (2006). Two newrobust genetic algorithms for the flowshop scheduling problem. **The International Journal the Management Science (Omega)**, v.34, p.461-476.

SCHUSTER, C.J.; FRAMINAN, J.M. (2003). Approximative procedures for no-wait job shop scheduling. **Operations Research Letters**, v.31, p.308-318.

SILVA, J.L.C. **O problema de sequenciamento aplicado na indústria téxtil**. Ceará, 1996. Dissertação – Programa de Mestrado em Matemática, Universidade Federal do Ceará, Fortaleza-Ceará, 1996.

SILVA, J.L.C.; SOMA, N.Y. Uma heurística para problemas de otimização combinatória permutacional. In: XXXIII Simpósio Brasileiro de Pesquisa Operacional, 2001, Camposdo Jordão-SP. XXXIII SBPO, 2001. p. 1298-1306.

SILVA, J.L.C.; SOMA, N.Y. Um método heurístico aplicado no problema de programação flow shop permutacional. In: XXXVIII Simpósio Brasileiro de Pesquisa Operacional, 2006, Goiânia. Anais do XXXVIII SBPO, 2006.

SILVA, J.L.C.; SOMA, N.Y. Um algoritmo genético híbrido construtivo polinomial aplicado ao flowshop scheduling problem. In: XIII Conferencia Latino-Ibero-Americana de Investigación de Operaciones, 2006, Montevideo. Anais do XIII CLAIO.

SISSON, R.L. (1959). Methods of sequencing in job shops - a review. **Operations Research**, v.7, p.10–29.

SURRY, P.D.; RADCLIFFE, N.J. (1996) **Inoculation to initialise evolutionary search**. In T.C.Fogarty (Ed.) (1996) Evolutionary Computing: AISB Workshop, Brighton, UK, April 1996; Selected Papers, Springer- Verlag, Berlin, 269-285.

TAILLARD, E. (1993). Benchmarks for basic scheduling problems. **European Journal of Operational Research**, v.64, p.278-285.

T'KINDT, V.; BILLAUT, J.-C. **Multi-criteria scheduling**. Springer-Verlag, Berlin. 1993.

WAGNER, H.M. (1959). An integer linear-programming model for machine scheduling. **Naval Research Logistics Quarterly**, v.6, p.131–140.

WANG, L.; ZENG,D.-Z. (2001). An effective hybrid optimization strategy for job-shop scheduling problemas. **Computers and Operations Research**, v.28, p.585-596.

ANEXO I – As melhores seqüências de trabalho obtidas pelo rAG no segundo experimento com o CPFSP para as instâncias de Taillard (1993).

tai001

3 17 9 15 14 8 16 13 1 2 6 10 7 20 12 11 19 4 5 18

tai002

14 10 6 20 12 2 8 18 4 3 7 9 11 16 19 15 1 13 5 17

tai003

3 19 11 6 2 4 13 16 9 8 1 15 10 14 17 18 7 20 5 12

tai004

9 16 8 14 4 18 13 17 12 19 3 6 11 15 10 2 1 5 7 20

tai005

3 12 20 1 8 5 10 13 9 7 15 19 18 4 17 16 11 2 14 6

tai006

2 20 14 17 13 5 3 8 1 11 6 12 7 16 18 10 15 9 19 4

tai007

10 2 15 16 1 14 18 17 13 7 4 20 19 5 12 9 11 3 8 6

tai008

12 6 2 14 4 3 1 17 16 9 7 13 20 11 10 18 15 5 19 8

tai009

4 8 16 7 12 10 14 5 13 15 2 20 1 19 3 9 6 11 18 17

tai010

7 19 15 13 5 9 11 12 1 2 18 20 17 16 6 10 3 14 8 4

tai011

18 5 12 20 11 15 4 1 2 16 10 7 6 14 9 3 17 19 13 8

tai012

9 13 4 6 17 19 3 20 14 12 18 8 10 16 11 15 1 7 2 5

tai013

4 19 17 3 15 1 18 5 6 7 9 2 12 10 11 13 20 16 14 8

tai014

18 4 17 19 11 5 6 13 20 10 9 1 3 12 16 2 15 7 14 8

tai015

16 17 14 10 3 9 11 18 13 6 1 12 4 15 7 5 19 2 20 8

tai016

18 16 11 12 20 15 10 9 8 3 7 4 2 1 5 14 6 13 19 17

tai017

4 7 10 11 19 6 15 5 3 16 20 8 17 13 18 1 9 12 14 2

tai018

7 8 16 13 20 10 4 17 3 5 19 15 14 2 9 6 11 18 12 1

tai019

14 6 8 11 12 20 1 16 7 10 9 13 3 18 17 4 2 19 5 15

tai020

12 16 6 3 14 19 5 2 9 11 18 20 17 4 13 15 1 10 7 8

tai021

19 3 5 10 16 14 7 9 15 1 6 18 4 17 2 11 12 13 8 20

tai022

8 5 9 13 17 19 11 20 18 1 10 4 7 6 14 2 15 12 16 3

tai023

2 8 20 17 6 3 14 19 7 11 1 12 9 16 15 18 10 5 13 4

tai024

14 4 11 18 2 6 8 5 20 1 7 15 9 19 10 3 17 16 13 12

tai025

18 9 7 15 12 8 6 14 4 13 19 16 17 20 11 5 2 1 10 3

tai026

11 13 1 20 2 16 3 4 14 8 19 18 15 6 5 17 9 7 10 12

tai027

14 4 15 17 9 7 3 20 10 11 6 12 5 8 18 16 1 2 19 13

tai028

2 16 1 15 9 6 13 12 8 19 18 5 11 7 20 4 17 3 10 14

tai029

7 13 19 8 4 15 6 14 12 11 18 17 20 10 16 1 5 9 2 3

tai030

7 9 14 6 2 16 4 20 13 11 8 5 17 19 18 1 3 15 12 10

tai031

10 24 36 37 17 39 49 20 38 46 3 12 31 50 40 42 48 23 32 41 44 7 6 18 16 13 2 26 22 33 35 19 30 1 11 21 25 43 8 4 47 34 5 28 15 29 27 45 14 9

tai032

50 49 38 15 23 47 42 18 3 6 34 36 10 5 2 14 11 8 44 29 7 41 21 33 37 22 45 13 30 12 16 43 20 32 39 26 1 28 19 9 17 4 31 25 27 40 46 48 24 35

tai033

8 22 15 23 12 18 27 37 21 36 49 2 3 28 25 35 39 26 1 29 48 30 31 6 14 34 16 19 10 45 9 17 43 7 11 46 41 24 40 5 42 33 20 4 38 47 32 13 50 44

tai034

42 22 12 6 49 9 46 43 31 30 28 2 27 37 44 50 23 15 18 14 47 21 35 16 24 25 41 45 40 13 3 36 19 39 32 7 4 1 34 38 26 11 10 29 5 20 48 8 17 33

tai035

46 48 5 45 50 32 7 34 13 9 42 44 27 10 3 30 19 4 22 37 6 33 29 28 18 40 14 1 43 24 23 21 8 49 31 16 39 35 12 25 36 11 15 26 17 38 20 41 2 47

tai036

4 21 1 29 22 12 31 5 25 9 47 32 41 46 27 33 24 40 28 19 43 34 20 36 6 50 3 37 26 8 48 42 11 39 44 10 2 14 16 30 15 23 18 35 38 49 7 17 13 45

tai037

27 28 22 16 25 5 34 30 40 15 4 50 19 38 47 43 45 42 41 8 36 13 37 14 24 23 1 49 21 39 6 12 17 26 7 20 18 44 2 10 32 31 3 29 46 48 9 35 11 33

tai038

34 17 4 21 7 23 40 35 47 30 22 13 38 2 36 10 31 46 37 3 15 26 9 20 1 33 27 50 14 25 18 24 29 43 41 49 5 8 19 39 12 32 11 16 42 48 6 28 44 45

tai039

17 29 14 13 10 46 9 24 50 8 12 45 1 3 16 21 44 47 36 31 28 7 34 32 4 26 40 37 38 39 6 27 18 22 41 2 49 5 35 11 20 15 23 30 48 25 33 19 42 43

tai040

50 44 30 19 43 23 31 6 20 36 21 18 33 49 42 7 17 12 45 1 34 48 3 26 37 29 28 35 39 47 9 22 25 32 27 8 11 16 13 41 24 38 46 40 15 10 4 2 5 14

tai041

42 44 33 20 34 6 10 1 43 7 19 17 8 47 18 22 48 39 32 35 26 24 13 30 37 14 31 36 46 21 41 28 49 2 12 4 38 3 25 29 16 11 9 40 15 23 5 45 50 27

tai042

35 47 40 5 29 7 17 49 1 22 46 10 26 3 24 48 42 33 28 23 14 11 50 31 44 18 2 30 45 15 9 38 32 13 6 37 20 19 36 43 41 27 34 21 8 25 12 39 16 4

tai043

24 4 28 31 12 8 15 16 27 3 32 11 46 39 18 22 36 9 10 7 21 23 26 2 1 5 17 40 48 30 14 49 37 19 45 25 35 34 6 41 29 43 33 47 50 38 42 20 13 44

tai044

20 10 19 9 44 29 5 37 18 40 16 34 13 7 33 39 23 11 25 36 35 30 21 8 22 17 45 41 38 12 26 24 50 32 4 31 48 3 27 46 42 28 1 2 49 6 14 15 43 47

tai045

6 50 34 10 42 48 12 30 33 25 31 1 35 9 46 7 20 23 29 45 36 11 3 49 4 13 19 18 44 21 43 39 15 40 26 37 41 24 5 16 14 22 32 27 17 38 47 28 8 2

tai046

3 42 33 20 45 31 23 26 40 39 47 27 34 35 38 28 44 19 37 15 24 5 11 9 16 41 46 22 8 50 4 7 17 25 14 6 43 49 12 21 32 10 30 1 29 18 2 36 48 13

tai047

6 41 27 33 48 17 40 46 13 38 22 44 20 3 28 25 11 35 23 31 36 18 21 26 34 2 1 30 15 45 12 8 7 10 9 43 37 49 32 16 50 39 4 29 42 24 19 5 14 47

tai048

28 31 32 41 21 13 17 34 15 35 1 6 3 44 37 45 33 42 23 5 38 43 40 39 36 11 7 30 14 20 26 29 22 19 8 4 25 48 2 12 24 9 47 49 50 18 46 16 10 27

tai049

33 30 25 50 45 40 44 6 42 31 39 12 10 38 13 28 21 35 19 49 32 20 1 5 47 7 37 26 46 8 22 48 11 27 34 2 36 23 41 14 43 24 9 15 29 18 16 4 3 17

tai050

49 8 38 10 14 21 19 4 41 37 15 27 9 42 45 13 24 23 2 12 28 39 44 29 40 33 46 32 6 1 48 16 26 34 47 7 36 30 5 22 50 20 18 43 17 3 11 31 25 35

tai051

37 27 8 44 43 20 15 39 34 31 41 47 32 30 38 33 17 50 3 26 40 24 29 9 12 36 13 49 28 48 22 21 7 10 35 6 42 1 18 25 4 46 11 45 14 2 5 19 16 23

tai052

32 49 8 39 31 40 33 7 38 20 41 30 16 22 10 36 3 14 23 6 29 21 24 50 5 4 13 35 17 47 42 11 45 1 12 26 18 9 15 37 2 28 48 44 27 25 19 46 34 43

tai053

24 4 28 11 25 7 31 2 1 3 19 39 26 10 41 35 12 18 6 30 14 21 23 49 36 8 15 16 27 46 5 45 9 17 40 33 34 29 47 38 44 13 43 48 20 42 50 32 22 37

tai054

1 19 47 14 8 13 34 38 9 44 48 40 25 37 46 39 11 24 50 27 18 12 45 5 20 43 42 7 30 4 21 15 35 31 17 16 41 32 2 33 6 10 26 22 3 49 28 29 23 36

tai055

23 25 11 7 36 43 49 46 37 28 40 20 15 50 21 3 39 38 10 34 44 24 48 45 32 17 30 47 16 35 13 1 33 19 42 9 29 27 41 22 6 14 12 18 8 2 4 31 5 26

tai056

14 33 5 49 37 11 25 42 4 39 29 12 47 26 6 24 9 15 38 3 13 27 17 35 2 19 31 46 8 1 18 50 45 21 41 36 44 32 48 16 40 23 7 34 22 28 30 20 43 10

tai057

4 35 5 15 42 48 17 9 23 38 13 45 2 30 44 27 36 19 31 11 46 28 1 20 24 6 25 37 22 32 41 14 34 12 10 40 16 26 43 18 21 47 33 8 29 7 3 49 50 39

tai058

33 39 1 7 19 42 28 12 2 29 32 30 8 48 27 26 35 9 3 20 14 40 11 13 31 6 50 23 22 37 45 38 15 36 18 16 5 41 49 21 47 43 25 34 24 46 10 17 44 4

tai059

44 29 35 15 11 31 9 14 50 1 27 37 43 45 3 38 23 47 5 20 12 16 22 41 30 21 28 8 2 24 17 19 34 42 26 33 36 40 25 4 48 6 13 10 32 46 18 39 7 49

tai060

38 11 18 10 1 16 39 43 15 48 49 20 42 12 3 23 2 5 28 30 31 19 46 22 29 21 37 50 27 13 26 25 17 45 4 33 6 35 41 14 8 32 40 7 44 36 34 24 47 9

tai061

10 93 46 5 40 16 66 55 84 19 59 24 65 12 82 72 56 62 7 14 77 26 96 33 34 88 83 71 58 15 92 61 35 20 60 29 30 50 42 80 78 3 36 64 95 23 68 1 85 39 28 21 97 99 11 63 79 47 87 74 8 13 44 2 98 76 69 53 32 49 38 37 6 51 94 31 45 89 4 75 17 27 43 91 73 67 90 41 81 52 54 18 86 22 25 57 48 9 70 100

tai062

69 79 33 46 45 99 10 15 83 88 98 77 65 16 86 92 25 53 93 8 20 6 61 100 39 1 60 90 75 56 26 34 58 52 89 5 80 67 82 19 76 29 31 37 68 78 14 57 81 43 66 24 70 36 97 91 54 48 38 72 17 42 40 12 85 47 96 7 22 55 73 28 50 51 32 27 95 63 84 64 30 41 13 9 59 23 94 62 35 21 18 71 3 87 4 74 11 49 2 44

tai063

55 68 23 42 1 6 43 58 25 77 57 51 11 60 89 17 86 71 90 88 31 94 82 30 32 46 7 64 41 13 28 2 96 12 10 61 34 40 62 21 79 95 26 97 18 100 45 48 16 54 44 24 83 80 99 69 56 49 19 75 53 76 22 5 36 65 59 70 47 52 39 38 14 93 15 50 85 67 91 9 8 35 4 33 37 87 92 98 74 20 78 63 27 3 29 73 81 84 66 72

tai064

96 51 21 16 37 55 56 18 22 64 43 14 80 58 85 52 83 57 38 93 67 20 28 9 88 45 78 81 73 54 70 31 95 65 77 82 32 63 42 35 79 23 89 91 98 10 61 66 49 59 26 7 69 19 3 33 4 60 71 46 84 40 2 94 72 36 27 5 11 13 50 17 44 75 97 76 62 39 6 68 8 99 1 90 92 15 25 100 86 87 47 41 12 29 34 24 74 48 30 53

tai065

1 2 74 68 10 16 50 98 77 12 33 79 41 57 75 91 58 86 70 94 18 24 82 29 76 5 11 64 28 87 72 51 47 20 39 61 4 52 37 83 66 78 7 100 53 17 8 22 92 38 69 62 25 97 56 96 84 54 60 27 44 63 99 67 48 40 80 34 19 93 46 30 81 73 15 88 71 23 3 26 13 85 90 55 21 59 35 6 32 89 9 65 31 42 43 49 36 95 14 45

tai066

4 54 83 21 57 51 47 8 65 36 29 61 80 45 96 99 71 53 37 95 69 1 85 72 27 90 49 3 19 77 94 56 7 2 16 33 76 39 44 86 70 50 100 26 92 20 88 14 30 58 78 34 23 67 32 91 98 6 5 93 75 38 46 9 73 22 81 87 79 42 40 97 18 48 15 31 55 68 59 28 62 25 11 74 89 17 12 84 64 41 10 60 35 63 24 52 66 13 43 82

tai067

28 79 13 20 2 5 6 35 50 27 64 24 81 98 14 12 62 71 89 16 92 66 15 40 67 91 75 93 11 4 76 29 77 22 7 80 19 85 69 1 38 96 10 21 72 61 42 3 53 99 100 45 46 41 68 70 54 30 86 26 90 51 49 44 8 9 82 95 32 57 18 94 43 59 84 58 87 33 31 74 78 83 39 47 36 23 63 60 97 37 52 34 73 48 65 56 17 25 88 55

tai068

42 56 2 17 64 41 90 21 97 40 95 87 98 74 1 52 15 25 53 29 86 82 73 67 30 14 70 80 48 27 75 9 59 22 85 79 43 34 68 76 8 36 6 83 91 49 31 19 65 54 32 13 5 11 55 99 72 45 18 69 84 77 58 3 35 100 47 26 96 7 33 46 62 23 16 89 88 20 92 78 94 28 50 12 44 4 81 61 24 66 71 93 39 10 57 51 60 63 37 38

tai069

70 24 72 47 60 21 40 97 4 73 9 29 28 58 90 48 63 6 43 100 95 75 98 82 10
84 57 2 53 8 17 19 51 81 56 49 44 59 52 34 79 78 80 12 33 42 15 69 36 16
1 32 99 85 68 66 35 23 22 45 65 50 5 71 93 39 11 83 54 20 92 67 31 13 46
87 18 86 61 55 96 74 14 3 27 88 38 94 62 7 37 91 30 76 64 77 89 26 41 25

tai070

2 20 27 46 70 74 62 83 31 68 47 92 75 37 10 90 23 57 1 58 11 28 88 54 7
32 50 12 65 26 29 94 67 76 51 42 15 55 93 5 80 49 69 72 22 16 38 81 35 18
60 100 98 44 9 48 43 45 95 82 33 87 4 79 64 6 99 53 13 66 56 77 34 91 97
61 19 85 25 73 71 21 84 40 39 63 30 52 8 36 14 59 41 78 86 89 96 3 17 24

tai071

58 64 70 21 15 29 26 45 72 12 36 77 40 74 49 2 5 61 28 82 17 14 81 62 78
69 30 59 87 95 91 24 98 53 99 47 19 83 96 34 56 94 71 46 63 13 27 43 93
84 39 60 8 31 90 18 51 7 88 100 16 86 89 11 20 33 9 79 42 80 55 38 35 25
54 73 92 32 66 97 48 50 6 85 3 4 76 41 67 10 57 68 22 1 23 75 65 52 44 37

tai072

24 99 73 64 3 16 75 28 81 76 51 4 21 80 8 46 12 66 10 78 11 91 36 43 15
69 49 54 98 18 83 40 7 38 56 2 72 87 95 39 6 31 61 60 62 9 19 47 63 13 35
77 1 53 26 44 68 79 71 86 23 89 58 74 25 41 52 82 5 20 57 34 37 65 27 50
33 30 84 97 85 14 42 70 55 17 59 96 92 94 88 93 48 45 90 67 32 22 29 100

tai073

45 25 87 23 58 64 4 16 99 57 39 94 12 42 74 96 72 66 97 20 80 29 63 92 56
34 27 93 50 48 40 9 38 83 32 13 44 41 81 6 61 21 55 73 51 65 15 18 28 5
89 24 54 88 76 14 2 37 62 52 1 36 85 98 70 67 86 91 17 35 31 11 79 10 100
33 60 19 8 69 46 82 26 75 77 59 53 22 78 90 3 43 95 7 84 30 68 47 71 49

tai074

24 76 85 95 46 61 2 90 77 62 30 79 63 98 68 23 97 80 39 55 28 19 14 56 32
99 52 69 26 94 64 83 81 12 5 40 7 58 13 72 8 22 33 15 6 35 75 70 20 78 59
31 34 10 50 37 65 74 60 53 21 71 49 82 47 67 42 27 41 29 9 3 84 44 1 43
17 54 18 16 51 100 11 86 93 96 92 48 25 88 73 66 91 57 45 36 89 38 87 4

tai075

83 79 65 95 80 90 66 25 93 50 100 19 33 46 5 9 75 56 45 47 26 78 91 55 76
7 21 40 94 71 59 11 72 92 62 31 52 28 99 20 63 51 88 36 70 1 48 53 86 17

57 32 87 44 18 13 35 43 15 24 14 29 74 38 82 22 2 42 81 8 98 6 49 12 37
68 73 89 16 67 3 64 84 97 58 10 77 85 39 4 96 34 54 27 61 30 23 60 41 69

tai076

78 79 5 76 92 36 19 46 41 98 48 75 44 28 45 38 9 57 6 20 4 95 40 77 18 64
49 70 22 59 65 3 47 90 97 66 30 56 81 71 37 33 34 26 7 32 24 42 43 15 53
52 72 60 99 63 12 67 39 29 58 54 8 74 23 94 50 35 96 25 2 10 21 91 87 11
83 88 86 68 55 69 14 80 93 100 62 1 27 85 84 31 73 89 17 61 51 13 16 82

tai077

76 1 13 56 53 67 59 24 52 14 22 65 39 27 47 74 87 28 5 69 32 54 86 4 83
42 49 6 38 7 12 35 33 64 79 34 46 80 40 23 58 16 26 71 91 97 11 9 21 96
15 55 48 89 3 61 19 17 31 8 68 37 43 44 29 45 25 92 18 99 50 70 20 95 51
98 2 94 75 63 41 85 88 62 66 100 57 93 82 36 60 90 72 84 81 77 10 73 78
30

tai078

48 63 67 17 90 81 80 10 59 55 71 3 33 97 76 50 12 86 40 20 85 47 11 14 66
96 41 70 31 73 56 9 28 45 93 78 21 77 53 1 74 30 72 15 94 65 34 98 19 22
13 83 75 95 51 2 61 68 38 36 89 57 69 92 5 25 6 24 26 27 23 8 44 18 49 62
54 87 52 43 32 99 88 29 46 79 84 64 58 82 42 60 35 16 7 91 39 100 4 37

tai079

91 54 92 64 43 19 67 23 86 29 21 42 18 62 4 10 1 79 100 81 85 74 9 46 75
73 57 36 95 98 2 94 20 68 25 76 3 48 38 34 53 15 16 41 27 80 26 17 96 60
44 65 83 84 66 51 52 24 49 56 45 97 77 31 87 5 58 33 99 61 35 47 30 8 89
39 69 7 71 14 72 88 32 70 11 90 50 37 40 12 22 28 63 82 13 78 55 93 59 6

tai080

84 57 48 97 81 71 99 2 14 9 78 15 68 63 100 16 32 64 19 47 62 34 6 87 52
75 17 8 40 89 88 54 66 76 36 21 30 20 80 42 67 38 29 25 55 10 58 11 41 53
93 90 86 96 98 91 73 77 69 56 22 44 3 24 79 82 94 4 70 31 7 28 18 37 1 35
59 39 12 26 72 5 27 85 13 23 92 50 74 49 45 43 65 95 46 33 60 83 51 61

tai081

1 59 36 94 46 3 93 31 39 61 12 4 5 20 19 75 89 74 80 58 97 13 14 47 38 51
37 92 78 90 2 62 79 27 49 77 41 88 73 50 69 98 44 57 11 82 25 9 54 65 60
32 85 83 81 16 56 6 28 55 66 21 70 52 63 23 45 67 91 29 26 96 15 95 7 87
84 24 30 72 68 86 34 43 71 17 18 99 76 22 33 10 100 48 35 42 64 40 8 53

tai082

50 76 14 29 98 5 6 15 65 24 38 90 48 9 99 23 72 40 26 83 81 51 16 4 44 62 30 46 55 73 28 63 87 92 58 64 96 27 13 97 61 89 34 54 22 84 95 43 75 70 20 80 56 74 57 94 39 12 42 25 66 49 100 69 82 91 35 31 3 36 2 37 32 52 10 47 79 68 19 78 88 67 86 85 17 18 1 60 7 93 71 11 33 8 77 41 21 53 59 45

tai083

83 37 97 82 75 30 55 62 14 65 45 95 2 96 74 21 28 19 94 87 10 4 9 86 63 22 20 39 76 78 31 48 81 85 47 70 88 58 17 16 3 52 89 69 60 93 8 49 71 12 27 79 18 100 15 38 99 92 64 41 40 56 90 11 23 91 53 46 25 36 26 73 1 13 59 32 34 43 80 77 84 67 54 61 24 42 51 44 66 98 7 29 50 33 57 35 72 68 6 5

tai084

36 80 33 57 89 84 52 21 12 58 25 67 51 26 43 91 66 77 30 7 100 79 15 61 62 27 14 34 11 45 41 17 82 48 39 6 40 73 90 3 2 29 37 74 42 78 4 99 72 10 35 94 18 65 49 5 44 98 23 63 60 68 93 81 19 55 47 28 22 46 86 38 95 50 85 20 9 24 96 92 59 64 83 97 1 32 16 76 8 70 71 13 31 87 75 54 56 69 88 53

tai085

51 49 33 91 36 67 13 15 71 30 99 38 93 7 19 61 54 77 79 9 44 27 23 39 75 98 72 10 12 83 5 26 100 17 60 11 3 81 74 73 22 18 68 20 55 66 85 76 87 58 32 2 16 95 4 31 34 48 24 56 45 28 50 94 90 21 84 80 62 86 78 43 52 37 8 97 64 14 96 42 69 65 53 35 57 70 40 82 1 46 88 92 29 41 6 89 63 25 59 47

tai086

31 12 83 32 96 73 33 89 92 1 78 27 80 65 29 94 54 50 67 70 6 61 63 30 88 79 43 60 36 47 51 59 93 42 90 24 44 18 87 2 45 15 4 38 76 21 22 58 84 23 39 7 66 48 5 69 82 37 56 19 62 75 91 55 71 11 34 97 64 14 10 17 16 99 57 98 3 20 40 85 77 26 86 41 25 28 49 52 8 95 74 9 68 46 53 13 72 100 35 81

tai087

95 50 41 33 28 25 75 27 88 85 94 62 93 14 21 9 1 45 16 32 49 52 44 20 79 26 19 64 60 5 80 78 22 35 76 38 92 83 3 100 65 18 56 71 15 86 96 30 2 57 66 37 68 13 53 12 72 24 84 23 29 82 36 40 59 43 51 34 98 39 4 74 87 89 61 7 11 73 6 67 58 97 8 42 55 99 54 77 90 47 91 46 17 70 31 48 63 10 81 69

tai088

70 22 87 90 2 73 3 15 4 96 28 39 6 32 48 79 21 99 46 50 8 43 86 30 60 27 64 95 74 54 35 53 66 61 68 25 1 17 10 45 75 23 41 29 89 33 20 97 26 42 52 88 63 31 69 19 62 12 71 24 59 72 56 84 38 14 78 76 77 82 83 65 94 91 85 11 36 67 58 7 37 81 57 51 93 40 47 55 34 49 9 100 13 92 16 5 18 98 44 80

tai089

66 44 7 2 76 71 58 90 80 84 99 27 92 21 88 15 29 63 6 89 10 68 74 22 75 64 42 24 35 3 36 95 77 51 4 56 1 13 28 17 34 47 60 16 43 62 50 81 37 91 23 53 93 65 59 30 11 8 78 20 73 94 41 46 26 14 83 61 79 85 87 25 96 57 100 69 31 72 48 86 18 33 70 98 45 5 55 54 19 9 40 67 82 38 12 97 52 39 32 49

tai090

11 48 28 73 44 53 67 20 39 57 18 2 70 43 88 7 65 100 31 8 42 89 85 66 46 21 35 15 6 22 47 49 82 79 81 14 36 94 59 41 91 45 55 63 75 30 23 17 64 90 27 24 52 77 62 34 56 84 5 26 4 98 37 25 50 12 33 86 99 69 3 61 80 58 76 95 16 96 54 1 38 83 71 13 19 9 68 32 78 72 51 60 87 40 92 10 97 74 29 93

tai091

73 29 28 94 65 15 188 148 30 33 124 71 190 172 132 104 110 49 34 61 3 99 169 160 176 77 147 125 168 183 161 23 98 52 149 156 1 90 146 182 113 142 18 92 187 19 97 8 106 24 63 81 47 136 197 67 57 103 127 6 173 83 75 109 178 78 39 137 152 163 25 36 195 151 164 170 60 167 191 85 87 69 16 158 180 93 129 76 91 135 200 68 64 43 22 198 2 9 101 186 35 50 159 118 111 46 20 134 196 138 42 45 66 13 128 10 54 114 184 21 162 102 27 41 192 74 105 116 107 4 17 84 55 123 157 89 140 155 150 53 100 14 171 70 174 194 154 12 193 58 130 88 31 199 86 165 37 120 117 80 175 48 189 115 108 112 26 72 79 139 82 95 51 144 122 38 121 181 166 59 11 62 143 177 141 126 145 40 56 179 44 32 7 5 131 153 185 119 96 133

tai092

31 166 42 150 161 70 44 115 101 163 11 29 48 119 14 97 63 188 61 26 13 130 94 96 65 57 9 62 107 181 133 25 112 5 34 74 200 145 186 182 15 98 86 109 45 33 183 8 27 170 104 137 156 149 64 40 21 55 88 143 155 17 7 73 187 54 164 158 92 69 174 140 22 68 24 75 66 198 178 138 185 132 171 162 121 194 125 28 114 2 117 139 192 141 127 91 195 76 99 146 49 190 50 152 179

59 4 41 53 38 103 144 32 77 47 71 30 78 82 111 175 122 134 39 191 58 3 89 147 142 37 126 124 18 46 189 19 148 87 129 157 60 131 167 23 196 10 93 128 16 118 20 84 67 105 136 123 173 1 83 160 110 197 165 135 102 120 6 79 199 56 172 168 81 12 113 106 193 36 51 180 85 80 52 184 108 90 159 153 95 151 72 154 169 116 35 43 176 177 100

tai093

97 52 95 83 92 32 2 39 133 166 157 62 183 72 93 147 42 102 187 57 159 129 134 104 43 135 25 189 128 192 46 148 146 96 49 36 103 58 153 154 78 89 18 91 119 180 15 44 1 126 198 191 173 29 20 51 84 8 68 100 186 179 101 94 41 16 53 82 176 69 184 64 60 150 174 61 3 24 50 88 4 140 14 10 86 70 143 188 145 116 163 19 113 48 110 155 17 118 121 108 33 98 28 125 66 34 123 181 27 115 77 141 85 38 73 182 162 99 87 12 47 71 142 136 190 160 120 22 21 131 164 5 63 106 90 23 9 59 132 199 74 31 151 124 144 65 193 194 112 169 171 6 56 107 117 158 80 40 137 200 149 35 54 7 172 196 161 67 177 26 114 197 30 75 195 138 185 167 122 76 168 139 79 156 105 152 81 175 170 111 55 165 127 130 109 45 11 13 37 178

tai094

160 196 90 192 161 56 131 158 147 71 144 136 11 187 50 130 120 61 134 133 92 103 195 176 128 142 24 80 118 86 74 36 189 111 69 163 32 5 115 184 116 105 145 199 173 76 10 125 15 23 137 82 21 106 30 197 112 165 96 13 64 99 17 75 59 55 153 25 35 156 200 27 151 126 178 119 188 194 38 180 58 174 67 185 170 91 132 42 3 179 78 190 44 43 198 181 183 108 19 100 127 94 104 157 45 29 57 124 33 51 140 16 39 9 149 97 162 123 98 155 41 8 150 117 62 159 83 182 186 110 89 65 20 146 88 79 121 102 68 66 193 4 177 122 47 107 168 26 37 40 84 52 70 167 139 135 54 77 28 73 154 169 6 63 129 46 171 1 101 12 143 113 7 191 138 34 166 93 22 164 53 2 18 81 87 14 48 141 60 72 95 85 109 148 175 114 152 172 49 31

tai095

198 46 189 90 86 186 188 124 32 73 14 101 172 193 108 181 58 92 132 105 11 111 29 106 33 52 84 22 131 149 174 74 89 144 96 95 161 165 80 71 42 109 48 57 168 91 175 17 72 3 98 93 117 195 55 125 26 126 63 35 157 185 51 192 61 37 171 70 128 103 40 164 67 60 64 127 79 135 12 110 155 2 50 182 154 7 25 5 76 143 147 153 123 194 4 178 114 83 75 104 170 47 15 139 184

140 173 179 38 113 130 85 183 13 6 160 68 82 120 180 150 54 8 152 156 16 122 66 141 190 36 137 177 43 97 166 169 176 196 163 44 81 191 65 10 136 28 151 118 27 121 187 1 87 148 62 34 23 200 59 102 138 21 20 45 99 112 115 77 145 41 49 39 53 129 162 119 133 69 31 159 94 146 30 18 107 100 56 197 116 24 78 167 9 88 134 142 19 158 199

tai096

147 85 13 36 102 47 197 168 118 6 78 134 89 126 178 195 52 69 131 9 143 183 39 107 53 172 3 146 190 164 132 55 158 51 40 138 23 29 25 41 65 186 122 72 114 57 180 67 12 174 103 37 20 156 116 149 1 26 93 99 152 198 95 98 11 10 79 42 61 106 76 60 73 135 18 128 173 181 115 187 145 80 162 142 108 192 133 109 148 159 94 28 68 167 14 2 170 92 169 104 120 137 110 35 101 44 160 166 48 175 184 64 111 63 157 71 91 153 112 136 191 17 165 15 54 141 46 155 75 163 125 200 59 171 30 74 82 16 84 151 90 22 49 66 8 43 62 176 34 130 70 196 185 50 161 199 150 123 38 127 21 129 117 121 81 27 31 33 189 179 140 24 5 32 87 77 83 119 7 86 56 88 45 58 154 144 177 96 97 182 193 139 105 19 4 188 113 124 100 194

tai097

135 198 147 80 117 197 127 28 38 173 179 41 98 145 15 97 56 126 63 36 182 11 88 57 169 75 94 192 34 191 73 196 151 39 71 18 159 52 187 60 50 161 132 87 184 163 115 26 123 141 119 49 107 168 177 23 165 170 85 45 51 4 91 143 139 35 93 46 104 29 76 111 78 108 20 53 77 133 12 90 155 180 17 47 83 194 13 181 24 40 10 144 72 6 32 27 131 152 190 19 105 42 25 112 14 178 162 160 22 54 120 74 66 189 193 43 5 33 110 121 95 116 100 167 70 86 101 154 109 58 195 129 79 122 142 37 153 176 7 199 140 102 185 149 148 136 31 114 62 61 3 99 183 2 186 150 103 157 172 118 188 128 171 89 16 96 65 174 84 130 134 1 48 67 21 175 44 9 8 81 166 106 30 158 113 200 137 124 82 92 68 138 156 164 69 146 125 55 64 59

tai098

153 105 82 154 112 180 90 185 23 1 114 87 43 38 179 101 73 116 94 128 141 8 24 26 20 45 167 34 2 4 83 103 18 176 28 25 75 181 125 118 163 12 92 192 120 148 108 84 156 168 65 137 79 49 165 152 46 170 166 131 50 55 117 194 191 175 85 74 182 119 132 197 70 58 16 190 29 110 37 183 76 135 66 93 177 19 10 150 151 122 31 88 14 69 144 3 71 107 121 162 96 200 17 145 109 11

143 106 80 7 61 52 98 99 104 59 97 89 102 169 47 91 184 9 195 44 178 139 155 67 62 173 22 146 27 129 51 147 63 186 54 95 13 158 123 78 172 81 196 30 193 53 130 36 134 56 68 188 111 60 174 199 35 72 64 40 33 100 187 142 86 32 161 42 41 159 15 77 140 133 126 5 189 48 149 115 6 164 124 160 157 39 198 171 127 57 138 136 113 21

tai099

97 42 20 65 144 177 180 39 165 18 78 91 3 168 19 145 22 138 175 151 188 73 125 30 119 191 46 158 187 126 71 157 156 174 100 26 141 186 195 24 17 189 76 28 58 184 88 5 45 37 77 142 133 148 9 159 143 69 154 196 15 111 149 40 176 118 12 178 110 137 198 161 72 199 62 127 49 112 128 106 164 181 48 68 81 89 35 116 132 101 59 122 136 34 82 194 115 84 107 93 105 64 121 56 67 55 21 74 83 200 163 124 86 113 103 170 185 95 51 53 135 44 104 70 11 123 167 146 61 60 130 169 4 140 25 129 171 57 13 109 33 54 2 99 102 152 147 172 94 160 90 6 50 16 179 29 197 85 31 8 108 23 190 162 134 47 7 150 32 38 87 80 36 10 173 63 66 92 41 183 139 1 27 166 75 43 193 52 114 182 79 96 192 98 155 117 131 14 153 120

tai100

148 177 85 143 138 149 46 87 43 103 180 174 188 94 101 1 199 135 89 76 14 69 127 109 57 81 132 38 97 52 182 28 164 168 12 65 152 58 51 2 155 157 184 140 123 83 24 74 3 145 197 193 178 116 158 130 31 119 63 5 66 108 139 196 195 50 9 92 48 141 166 8 136 95 40 111 60 23 176 19 121 39 73 86 190 181 18 22 170 25 147 172 104 131 165 10 186 29 53 4 179 106 154 133 162 187 151 6 134 98 198 161 32 68 77 107 120 45 146 173 82 169 79 21 36 100 72 102 67 80 167 7 142 64 11 117 47 96 124 71 126 153 112 62 20 105 122 90 93 113 35 200 192 54 88 194 128 185 16 42 61 183 150 55 163 33 84 41 70 15 175 34 110 75 114 49 13 144 137 56 159 78 59 129 125 37 27 99 189 156 115 17 26 160 118 44 91 191 30 171

tai101

83 95 151 198 76 170 29 21 193 138 20 190 174 23 40 90 75 86 183 128 60 65 97 92 62 82 49 88 22 195 162 140 167 131 89 107 194 78 192 96 61 10 159 185 109 166 163 43 145 64 19 152 33 142 155 63 28 24 47 143 26 160 17 50 91 99 146 39 113 132 69 187 120 46 122 25 31 153 171 45 41 141 154 164 32 53 175 55 14 66 67 6 12 77 196 112 15 197 30 44 181 124 121 94 8 9 178

111 5 93 144 182 7 36 126 79 147 80 136 57 81 34 189 52 68 172 85 158 101 103 165 38 115 73 118 114 3 186 74 179 176 108 48 18 13 16 71 58 148 119 84 1 116 37 102 180 130 54 169 184 156 110 191 11 87 56 188 125 133 59 105 135 168 173 106 150 100 4 149 161 127 98 27 134 70 35 129 72 157 139 51 104 123 117 2 42 177 200 199 137

tai102

56 132 78 26 37 118 43 97 50 92 117 12 83 49 177 172 165 157 86 101 25 73 140 87 136 164 67 69 60 168 103 11 124 2 173 149 179 21 53 3 18 129 114 180 44 151 160 144 159 119 52 94 155 35 146 20 198 81 110 127 90 191 68 111 138 126 167 197 29 24 96 184 135 58 125 200 131 98 88 169 64 32 70 8 45 181 147 142 175 10 115 62 189 51 193 42 113 162 34 123 84 120 143 102 152 57 48 182 15 33 161 27 39 61 22 77 4 47 13 171 99 75 79 38 41 46 156 134 59 121 63 141 190 7 95 72 17 150 174 170 55 28 76 109 82 130 1 128 16 188 6 19 40 154 112 194 65 196 105 85 80 36 153 145 5 106 9 93 186 23 163 54 30 100 139 137 148 183 178 107 176 108 116 122 71 66 187 195 185 104 14 199 158 192 91 74 166 133 89 31

tai103

179 30 132 127 177 111 178 20 187 44 189 184 82 33 162 90 8 152 73 106 46 194 182 61 168 133 159 110 88 58 6 4 134 147 119 101 40 35 125 53 129 81 5 121 59 64 34 139 107 43 67 87 51 38 171 97 144 9 163 14 185 145 72 128 96 195 198 167 123 108 23 165 32 126 112 156 160 192 86 122 153 85 76 80 68 148 186 105 155 151 93 117 138 77 25 115 1 199 54 22 200 143 48 95 173 15 13 146 37 31 183 102 136 158 161 57 49 176 47 78 113 84 169 41 98 174 45 10 166 16 75 130 65 170 193 157 7 70 21 114 109 62 120 55 12 2 50 69 164 24 66 3 149 154 191 188 42 11 27 181 135 141 91 26 29 137 180 116 131 52 103 100 124 19 71 196 190 118 17 104 172 83 140 99 175 60 150 92 74 39 94 18 56 63 28 197 89 142 79 36

tai104

66 43 49 168 15 188 73 148 28 160 3 137 1 33 129 78 39 30 29 99 60 123 183 85 87 195 120 146 110 83 42 158 136 164 151 21 64 19 18 16 175 154 4 26 199 180 93 101 139 40 88 114 24 81 77 149 191 57 71 109 165 20 8 84 147 27 17 178 157 52 23 172 134 6 14 41 196 45 65 124 98 105 194 141 82 55 7 132 2 95 46 89 169 187 75 106 122 68 47 36 155 182 37 103 100 170

186 35 91 54 70 128 143 31 72 5 63 92 50 193 200 184 127 90 173 171 38 104 69 107 130 118 53 140 163 190 176 67 167 142 34 125 152 198 150 96 74 189 48 12 177 166 112 179 44 145 86 144 59 174 116 121 117 80 192 9 10 13 11 108 32 102 181 153 185 56 62 126 159 79 76 25 161 113 197 51 58 162 135 97 94 156 61 138 115 22 111 133 131 119

tai105

162 83 31 86 168 27 70 112 193 110 8 53 71 172 23 46 17 165 127 148 195 131 41 145 36 149 67 137 196 101 55 6 156 45 128 50 5 37 105 98 100 134 11 97 56 38 3 63 142 40 194 118 25 72 28 43 77 135 10 190 133 102 130 74 117 129 61 66 186 185 158 107 163 29 132 178 85 76 180 174 120 106 124 22 138 52 2 1 154 175 95 68 93 199 99 26 103 150 4 147 116 155 123 125 44 111 80 87 58 167 200 82 69 59 157 113 164 152 64 136 139 108 19 47 9 177 114 30 176 104 51 18 32 141 169 34 24 144 159 160 75 187 48 151 91 88 35 49 79 90 189 197 78 119 84 140 153 181 33 81 15 183 166 57 126 146 96 14 192 89 12 188 182 20 73 7 16 115 60 143 184 122 62 198 191 54 92 39 171 173 13 121 94 21 42 65 109 179 170 161

tai106

152 170 78 163 190 121 98 195 39 81 142 101 141 178 11 76 99 119 160 157 117 88 198 97 16 62 51 94 103 87 3 126 33 82 29 127 32 23 38 60 145 96 75 92 175 185 107 122 128 44 34 192 139 110 159 80 200 137 115 196 169 13 179 21 151 158 84 42 35 85 172 149 40 164 45 95 77 104 168 197 12 74 56 72 58 90 79 129 156 43 50 146 30 10 183 116 31 15 61 52 68 83 194 109 176 53 171 177 47 89 18 138 123 165 130 199 113 24 186 106 65 131 41 91 111 132 136 14 155 57 63 112 144 118 64 182 27 148 93 184 1 187 17 49 19 37 2 71 180 59 67 4 140 5 102 20 86 9 150 54 191 181 154 147 153 70 173 7 36 125 105 8 108 46 174 188 124 120 28 100 25 6 189 73 193 135 48 133 66 69 167 114 143 55 162 22 26 161 166 134

tai107

200 190 168 29 146 126 59 173 64 27 35 68 181 164 14 154 176 7 69 15 71 46 103 30 199 192 28 12 131 175 9 182 17 91 98 162 52 179 21 45 105 3 186 196 48 22 122 10 44 65 23 101 194 166 153 90 156 111 198 177 133 160 42 36 92 152 25 1 41 119 169 67 31 112 76 114 72 147 130 6 100 75 83 47 163 138 165 155 49 5 50 115 106 104 18 124 110 125 193 74 143 88 158 4 195

150 139 123 121 132 97 120 109 99 2 116 108 189 187 13 34 26 161 66 140 117 149 86 70 188 174 159 96 77 19 63 142 102 32 82 148 141 53 11 113 171 84 60 151 37 73 94 54 184 136 89 40 80 85 167 128 157 81 51 38 129 56 134 127 8 185 118 183 58 191 180 93 197 145 95 57 62 178 144 24 55 79 61 135 33 172 39 78 137 170 43 16 87 107 20

tai108

196 36 25 122 16 10 151 83 50 125 81 190 132 30 172 121 157 97 150 69 106 61 20 63 176 58 141 184 119 115 193 48 162 165 37 137 47 179 171 9 80 152 70 22 109 199 126 73 31 93 90 91 131 89 44 24 68 104 67 146 62 143 64 28 194 178 14 2 189 53 27 82 175 112 11 6 111 164 155 26 117 43 154 76 38 198 186 3 60 35 33 160 94 65 32 4 128 127 95 192 191 52 114 39 54 5 133 29 124 161 136 105 123 42 102 138 173 8 149 174 147 45 116 23 85 110 139 177 129 153 49 92 182 148 17 72 86 169 100 77 185 40 46 71 197 21 74 7 96 107 88 135 59 134 183 158 101 12 187 103 57 200 144 99 156 51 108 167 120 18 55 87 75 98 180 130 118 188 41 140 145 113 13 181 34 15 195 56 159 163 66 78 19 168 79 1 170 142 166 84

tai109

190 10 148 199 151 166 25 113 160 54 50 70 100 106 19 77 36 55 29 61 74 154 116 153 86 72 101 136 111 28 62 142 80 24 134 94 124 1 20 8 58 16 30 135 14 2 92 192 37 84 173 197 67 76 22 117 6 57 174 158 89 105 78 93 83 46 146 96 69 168 172 186 11 147 71 17 123 145 130 98 144 110 64 170 21 155 102 189 51 161 99 40 162 149 163 68 108 122 26 159 59 194 43 179 90 52 129 15 103 81 23 175 125 121 45 91 41 79 138 107 140 156 169 143 182 13 114 167 133 126 56 115 7 187 164 66 127 5 3 120 165 87 131 73 34 65 178 95 157 32 185 4 75 42 112 183 137 150 63 85 104 27 200 60 176 181 82 118 132 53 31 171 12 47 48 193 139 128 180 152 18 141 198 38 184 177 88 196 39 97 35 191 9 33 119 188 195 109 44 49

tai110

130 15 116 7 196 142 154 197 77 180 149 47 103 100 83 12 151 59 13 141 94 198 50 131 124 104 126 112 91 177 81 52 65 14 150 172 75 60 189 191 127 55 135 160 163 56 133 139 67 64 192 144 199 121 140 178 173 53 98 48 155 183 122 128 106 21 101 87 167 54 93 40 145 114 27 28 66 123 161 9 132 42 118 108 29 22 169 76 97 17 190 32 25 46 79 57 1 2 61 147 164 158 88 95 49

38 200 153 43 34 111 31 51 20 119 102 99 134 69 181 62 115 73 92 187 113
35 194 4 174 8 175 18 159 157 120 162 10 86 109 74 68 6 11 182 85 23 170
184 82 129 138 90 44 195 45 71 166 165 156 117 19 168 179 125 33 5 39 37
96 105 3 41 80 30 137 152 136 143 63 26 78 148 176 89 24 72 107 171 36
110 193 58 186 188 185 146 84 70 16

ANEXO II – As melhores seqüências de trabalhos obtidas pelo rAG no quarto experimento com o CPFSP para as instâncias de Reeves (1995) e Heller (1960).

rec01

6 2 15 13 11 7 20 4 17 1 5 10 9 8 18 14 12 16 3 19

rec03

2 9 5 14 7 8 16 10 18 4 3 1 17 15 12 13 11 19 20 6

rec05

12 19 11 9 6 1 8 13 14 2 20 3 5 18 4 15 10 7 16 17

rec07

10 13 11 4 9 12 3 18 16 8 6 7 15 5 17 1 2 19 14 20

rec09

16 15 20 17 14 18 11 1 12 6 7 5 13 8 10 9 19 3 2 4

rec11

16 4 2 20 18 7 14 9 8 17 10 12 13 19 11 15 1 3 5 6

hel2

13 1 2 9 10 4 20 8 19 7 14 11 3 6 15 5 16 17 18 12

rec13

4 3 14 11 17 8 12 2 10 15 7 6 16 20 18 1 13 19 9 5

rec15

12 1 16 9 13 2 5 15 6 19 10 20 17 14 11 8 3 18 4 7

rec17

20 12 18 2 17 13 19 4 14 7 3 10 11 8 1 6 9 16 15 5

rec19

5 7 21 17 20 6 13 10 15 29 22 14 11 2 1 3 4 12 27 23 8 24 9 19 30 26 25 16 18 28

rec21

23 12 14 7 13 17 1 24 8 26 16 20 28 29 18 5 11 19 10 9 4 6 15 2 27 25 30 21 3 22

rec23

3 24 29 1 5 2 21 20 4 16 14 9 19 26 22 28 15 8 30 23 10 18 13 17 25 11 6 7 27 12

rec25

29 3 24 20 23 6 16 21 11 2 28 30 14 15 22 25 10 4 5 7 12 1 9 19 8 18 27 17 26 13

rec27

17 19 25 24 1 9 5 30 27 14 29 18 10 4 11 3 23 16 22 20 13 7 12 15 6 8 28 2 21 26

rec29

15 29 25 26 2 7 11 4 23 6 10 12 30 1 22 8 9 20 16 17 14 24 13 5 27 28 3 21 18 19

re31

34 40 46 6 23 48 8 44 26 27 50 38 13 2 24 47 35 32 10 37 25 28 17 14 22 29 31 11 42 15 9 36 30 49 7 12 43 21 20 4 5 1 33 3 16 41 39 18 45 19

rec33

47 18 2 36 22 34 39 38 9 28 41 42 5 50 10 1 6 7 11 26 21 8 13 48 20 49 16 15 27 19 14 44 31 3 37 25 43 32 29 46 33 30 12 35 4 17 40 45 24 23

rec35

25 6 27 2 38 35 13 36 42 20 18 41 10 39 50 32 48 31 43 1 5 14 40 17 3 44 30 26 4 33 15 8 23 19 9 12 45 37 21 28 7 47 34 24 11 46 29 49 16 22

rec37

41 19 18 50 1 63 40 75 44 48 67 53 56 20 61 43 29 28 9 7 32 12 65 66 25 58 60 2 57 16 36 73 55 49 42 3 31 46 69 4 51 74 45 11 64 59 13 34 17 39 47 26 15 68 52 54 10 33 72 6 22 8 71 5 27 30 21 14 62 35 24 38 23 70 37

rec39

24 20 47 40 63 56 45 68 23 12 59 16 42 19 57 44 43 48 32 22 11 55 3 52 54 34 15 61 66 46 13 4 58 38 10 31 18 35 49 65 28 9 30 29 69 37 14 21 1 73 72 39 71 27 74 41 51 6 5 2 25 53 50 17 64 36 26 8 33 67 75 62 60 70 7

rec41

30 68 69 28 7 44 29 72 19 35 52 6 24 54 50 65 23 5 64 67 34 63 41 66 37 59 13 48 3 58 75 71 51 74 18 42 26 33 36 46 9 10 25 60 22 31 47 43 14 8 11 45 20 73 4 39 56 17 2 32 55 49 21 40 53 57 61 70 16 12 27 62 1 38 15

hel1

13 37 63 74 98 2 87 48 82 53 32 43 25 24 4 80 67 21 40 58 5 71 52 92 14 94 55 16 17 84 76 15 1 22 90 30 91 65 38 78 72 23 62 41 6 59 51 60 64 11

33 3 26 9 93 42 68 27 95 46 86 35 44 75 99 79 97 49 88 47 73 70 57 50 96
10 77 29 66 100 89 20 8 85 39 61 36 34 19 28 7 31 12 45 69 54 81 18 83 56