



Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Instituto Politécnico

Tatiana Balbi Fraga


**Proposição e análise de modelos híbridos para o problema de
escalonamento de produção em oficina de máquinas**

Nova Friburgo

2010

Tatiana Balbi Fraga

**Proposição e análise de modelos híbridos para o problema de
escalonamento de produção em oficina de máquinas**



Tese apresentada como requisito parcial para
obtenção do título de Doutor, ao Programa de
Pós-Graduação em Modelagem Computacional,
do Instituto Politécnico, da Universidade do
Estado do Rio de Janeiro.

Orientador: Prof. Antônio José da Silva Neto

Co-Orientador: Prof. Francisco José da Cunha Pires Soeiro

Orientador no INSA de Rouen do doutorado sanduíche: Prof. José Eduardo Souza de Cursi

Nova Friburgo

2010

CATALOGAÇÃO NA FONTE
UERJ/REDE SIRIUS/BIBLIOTECA CTC/E

F811 Fraga, Tatiana Balbi.

Proposição e análise de modelos híbridos para o problema de escalonamento de produção em oficina de máquinas / Tatiana Balbi Fraga. - 2010.
139 f. : il.

Orientadores: Antônio José da Silva Neto e Francisco José da Cunha Pires Soeiro.

Tese (Doutorado) - Universidade do Estado do Rio de Janeiro, Instituto Politécnico.

1. Escalonamento de produção – Teses. 2. Algoritmo – Teses. 3. Modelagem matemática – Teses. 4. Heurística – Teses. I. Silva Neto, Antônio José da. II. Soeiro, Francisco José da Cunha Pires. III. Universidade do Estado do Rio de Janeiro. Instituto Politécnico. IV. Título.

CDU 519.712

Autorizo, apenas para fins acadêmicos e científicos, a reprodução total ou parcial desta tese, desde que citada a fonte.

Assinatura

Data

Tatiana Balbi Fraga

**Proposição e análise de modelos híbridos para o problema de
escalonamento de produção em oficina de máquinas**

Tese apresentada como requisito parcial para
obtenção do título de Doutor, ao Programa de
Pós-Graduação em Modelagem Computacional,
do Instituto Politécnico, da Universidade do
Estado do Rio de Janeiro.

Aprovado em 26 de março de 2010.

Banca examinadora:

Prof. Antônio José da Silva Neto, Ph.D. (Orientador)
Instituto Politécnico - UERJ

Prof. Francisco José da Cunha Pires Soeiro, Ph.D. (Orientador)
Universidade do Estado do Rio de Janeiro

Prof. José Eduardo Souza de Cursi, Ph.D.
Institut National des Sciences Appliquées de Rouen

Prof. Luiz Biondi Neto, D.Sc
Universidade do Estado do Rio de Janeiro

Prof. Edson Luiz Cataldo Ferreira, D.Sc.
Universidade Federal Fluminense

Prof. Horácio Hideki Yanasse, Ph.D.
Instituto Nacional de Pesquisas Espaciais

Nova Friburgo

2010

AGRADECIMENTOS

Agradeço a todos que contribuíram com este trabalho, em especial agradeço:

- À minha mãe que sempre esteve presente;
- Ao meu querido orientador Prof. Antônio José da Silva Neto que fez germinar em mim a arte da pesquisa e se tornou um exemplo de caráter, sempre íntegro, profissional e humano;
- Ao meu Co-orientador Prof. Francisco Soeiro cujo auxílio foi imprescindível;
- Ao Prof. Eduardo De Cursi cuja orientação foi de ajuda inestimável;
- À CAPES pelo apoio financeiro;
- À Deus por tudo que ele tornou possível.

RESUMO

FRAGA, Tatiana Balbi. *Proposição e análise de modelos híbridos para o problema de escalonamento de produção em oficina de máquinas*. 2010. 139 f. Tese (Doutorado em Modelagem Computacional) – Instituto Politécnico, Universidade do Estado do Rio de Janeiro, Nova Friburgo, 2010.

Nas últimas décadas, o problema de escalonamento da produção em oficina de máquinas, na literatura referido como JSSP (do inglês *Job Shop Scheduling Problem*), tem recebido grande destaque por parte de pesquisadores do mundo inteiro. Uma das razões que justificam tamanho interesse está em sua alta complexidade. O JSSP é um problema de análise combinatória classificado como NP-Difícil e, apesar de existir uma grande variedade de métodos e heurísticas que são capazes de resolvê-lo, ainda não existe hoje nenhum método ou heurística capaz de encontrar soluções ótimas para todos os problemas testes apresentados na literatura. A outra razão baseia-se no fato de que esse problema encontra-se presente no dia-a-dia das indústrias de transformação de vários segmentos e, uma vez que a otimização do escalonamento pode gerar uma redução significativa no tempo de produção e, consequentemente, um melhor aproveitamento dos recursos de produção, ele pode gerar um forte impacto no lucro dessas indústrias, principalmente nos casos em que o setor de produção é responsável por grande parte dos seus custos totais. Entre as heurísticas que podem ser aplicadas à solução deste problema, o Busca Tabu e o Multidão de Partículas apresentam uma boa performance para a maioria dos problemas testes encontrados na literatura. Geralmente, a heurística Busca Tabu apresenta uma boa e rápida convergência para pontos ótimos ou sub-ótimos, contudo esta convergência é frequentemente interrompida por processos cíclicos e a performance do método depende fortemente da solução inicial e do ajuste de seus parâmetros. A heurística Multidão de Partículas tende a convergir para pontos ótimos, ao custo de um grande esforço computacional, sendo que sua performance também apresenta uma grande sensibilidade ao ajuste de seus parâmetros. Como as diferentes heurísticas aplicadas ao problema apresentam pontos positivos e negativos, atualmente alguns pesquisadores começam a concentrar seus esforços na hibridização das heurísticas existentes no intuito de gerar novas heurísticas híbridas que reúnam as qualidades de suas heurísticas de base, buscando desta forma diminuir ou mesmo eliminar seus aspectos negativos. Neste trabalho, em um primeiro momento, são apresentados três modelos de hibridização baseados no esquema geral das Heurísticas de Busca Local, os quais são testados com as heurísticas Busca Tabu e Multidão de Partículas. Posteriormente é apresentada uma adaptação do método Colisão de Partículas, originalmente desenvolvido para problemas contínuos, onde o método Busca Tabu é utilizado como operador de exploração local e operadores de mutação são utilizados para perturbação da solução. Como resultado, este trabalho mostra que, no caso dos modelos híbridos, a natureza complementar e diferente dos métodos Busca Tabu e Multidão de Partículas, na forma como são aqui apresentados, da origem à algoritmos robustos capazes de gerar solução ótimas ou muito boas e muito menos sensíveis ao ajuste dos parâmetros de cada um dos métodos de origem. No caso do método Colisão de Partículas, o novo algoritmo é capaz de atenuar a sensibilidade ao ajuste dos parâmetros e de evitar os processos cíclicos do método Busca Tabu, produzindo assim melhores resultados.

Palavras-chave: Escalonamento da produção. Busca Tabu. Multidão de Partículas. Colisão de Partículas. Hibridização. Otimização.

ABSTRACT

In recent decades, the Job Shop Scheduling Problem (JSSP) has received great attention of researchers worldwide. One of the reasons for such interest is its high complexity. The JSSP is a combinatorial optimization problem classified as NP-Hard and, although there is a variety of methods and heuristics that are able to solve it, even today no method or heuristic is able to find optimal solutions for all benchmarks presented in the literature. The other reason builds on noted fact that this problem is present in day-to-day of industries of various segments and, since the optimal scheduling may cause a significant reduction in production time and thus a better utilization of manufacturing resources, it can generate a strong impact on the gain of these industries, especially in cases where the production sector is responsible for most of their total costs. Among the heuristics that can be applied to the solution of this problem, the Tabu Search and the Particle Swarm Optimization show good performance for most benchmarks found in the literature. Usually, the Taboo Search heuristic presents a good and fast convergence to the optimal or sub-optimal points, but this convergence is frequently interrupted by cyclical processes, offset, the Particle Swarm Optimization heuristic tends towards a convergence by means of a lot of computational time, and the performance of both heuristics strongly depends on the adjusting of its parameters. This thesis presents four different hybridization models to solve the classical Job Shop Scheduling Problem, three of which based on the general schema of Local Search Heuristics and the fourth based on the method Particle Collision. These models are analyzed with these two heuristics, Taboo Search and Particle Swarm Optimization, and the elements of this heuristics, showing what aspects must be considered in order to achieve a best solution of the one obtained by the original heuristics in a considerable computational time. As results this thesis demonstrates that the four models are able to improve the robustness of the original heuristics and the results found by Taboo Search.

Keywords: Scheduling problem. Tabu Search. Particle Swarm Optimization. Particle Collision. Hybridization. Optimization.

SUMÁRIO

1	INTRODUÇÃO	8
1.1	Algoritmos exatos	9
1.2	Algoritmos de aproximação	10
1.2.1	<u>Heurísticas de Busca Local</u>	11
1.2.2	<u>Hibridização de algoritmos</u>	15
1.3	Objetivos específicos desta tese	16
2	PROBLEMA DE ESCALONAMENTO DA PRODUÇÃO EM OFICINA DE MÁQUINAS	18
2.1	Modelagem matemática	21
2.1.1	<u>Modelagem clássica</u>	21
2.1.2	<u>Modelagem disjuntiva</u>	22
2.1.3	<u>Modelagem Inteira Mista</u>	23
2.2	Problemas padrões apresentados na literatura	23
3	SOLUÇÃO DO PROBLEMA DE ESCALONAMENTO DA PRODUÇÃO EM OFICINA DE MÁQUINAS	25
3.1	Busca Tabu	25
3.1.1	<u>Método de geração de vizinhança N5</u>	26
3.2	Otimização por Multidão de Partículas	29
3.2.1	<u>Código de Processamento de Serviços</u>	34
3.2.2	<u>Operador de cruzamento C1 com um ponto de corte flutuante</u>	38
3.2.3	<u>Operador de mutação M7</u>	39
3.3	Modelos de hibridização baseados nas Heurísticas de Busca Local	41
3.3.1	<u>Aplicação Híbrida Sucessiva</u>	43
3.3.2	<u>Vizinhança Híbrida</u>	44
3.3.3	<u>Vizinhança Híbrida Melhorada</u>	45
3.4	Colisão de Partículas Similar	47
4	RESULTADOS E DISCUSSÕES	52
4.1	Análise de sensibilidade do <i>makespan</i> ao valor dos parâmetros “número máximo de iterações sem melhora do <i>makespan</i>” e “tamanho da lista tabu” para o método Busca Tabu.	54
4.2	Análise de convergência	58
4.2.1	<u>Resultados da análise de convergência para o modelo de hibridização Aplicação Híbrida Sucessiva</u>	58
4.2.2	<u>Resultados da análise de convergência para o modelo de hibridização Vizinhança Híbrida</u>	60
4.2.3	<u>Resultados da análise de convergência para o modelo de hibridização Vizinhança Híbrida Melhorada</u>	62

4.2.4	<u>Resultados da análise de convergência para o modelo de hibridização Colisão de Partículas Similar</u>	65
4.2.5	<u>Análise dos resultados encontrados com base no operador Busca Tabu Paralela</u>	66
4.2.6	<u>Análise de convergência para os problemas FT10 e ABZ5 a ABZ9</u>	67
4.3	Análise de sensibilidade do <i>makespan</i> ao valor do parâmetro “tamanho da lista tabu” do método Busca Tabu para os quatro modelos de hibridização.	69
4.4	Análise de sensibilidade do <i>makespan</i> ao valor do parâmetro “número máximo de iterações sem melhora do <i>makespan</i>” do método Busca Tabu para os quatro modelos de hibridização.	71
4.5	Análise de sensibilidade do <i>makespan</i> ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” do operador Otimização por Multidão de Partículas Similar para os modelos de hibridização Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada.	72
5	CONCLUSÕES E TRABALHOS FUTUROS	75
	REFERÊNCIAS	77
	APÊNDICE 1 – Problemas utilizados para testes	81
	APÊNDICE 2 – Tabelas de resultados	86

1 INTRODUÇÃO

Segundo mostra uma pesquisa realizada pela Confederação Nacional da Indústria (2006) em conjunto com o SEBRAE, a competitividade da indústria brasileira se ampliou no início desta década. De acordo com esse estudo, tal fato resultou dos esforços empreendidos pelas empresas industriais no sentido de elevar a qualidade de seus produtos e a produtividade de seus processos de fabricação. As empresas brasileiras expostas a uma acirrada concorrência, que resultou da abertura comercial e da maior integração à economia internacional, têm buscado continuamente soluções para aumentar a sua competitividade e, como resposta a mudança de um exigente mercado consumidor, muitas dessas empresas passaram a produzir lotes cada vez menores de uma ampla faixa de produtos (variedade), gerando um aumento da necessidade de planejamento. Nesse cenário o escalonamento da produção, que tem como foco principal a alocação de recursos finitos para as tarefas buscando melhorar a utilização destes recursos, tem assumido cada vez mais um papel de importância nas prioridades estratégicas. Fraga (2006) observou em sua dissertação de mestrado a relevância do escalonamento para pequenas e médias empresas, como no caso das empresas do setor de confecções do município de Nova Friburgo, onde o mesmo é geralmente realizado de forma precária, com base no conhecimento empírico e sem o auxílio de qualquer ferramenta que possa melhorar a eficiência da produção. Através do escalonamento é possível otimizar a utilização dos diversos recursos e, no caso de empresas como as do segmento de confecções onde grande parte dos custos é gerada pelo setor de produção (energia, mão de obra, manutenção de equipamentos, etc), um escalonamento bem planejado pode aumentar significativamente a taxa de lucro, sem contar outras vantagens como a definição mais precisa de prazos de entrega, previsão de despesas e de consumo de matéria prima entre outros recursos de produção, além da possibilidade de planejamento do aproveitamento dos tempos ociosos de máquinas para manutenção de equipamentos, folgas programadas ou mesmo adiantamento de tarefas.

O problema de escalonamento da produção vêm sendo estudado desde o início da década de 50 quando Johnson (1954) desenvolveu um algoritmo de otimização para um problema do tipo “*flow shop*” que considerava apenas duas máquinas. Posteriormente este problema ganhou novas formas, dando origem a uma série de “problemas padrões de escalonamento”. Dentre estes o problema de escalonamento da produção em oficina de

máquinas, no inglês referido como *job-shop scheduling problem* (JSSP), ganhou grande destaque na literatura por ser considerado não apenas como um dos mais complexos problemas de escalonamento da produção, mas como um dos mais difíceis problemas dentro do campo da análise combinatória (SHA; HSU, 2006). Isso pode ser ilustrado pelo fato de que um problema relativamente pequeno com 10 serviços e 10 máquinas, proposto por Fisher e Thompson (1963), restou sem solução por mais de um quarto de século e, atualmente, existe ainda uma grande dificuldade para encontrar-se soluções ótimas para problemas testes com 20 serviços e 20 máquinas, em tempos de processamento praticáveis (ZHANG et al., 2008). Essa dificuldade pode ser explicada pelo fato de que, conforme comprovado por Lawer (1989), o JSSP é classificado como NP-Difícil e, portanto, não pode ser resolvido de forma ótima em tempo polinomial por nenhum método de otimização conhecido.

Apesar do JSSP ser apenas uma simplificação dos problemas reais de escalonamento que surgem em alguns processos de manufatura, sua resolução pode auxiliar em muito na tomada de decisão concernente à produção e consiste em um importante passo para a solução de problemas mais complexos e que são mais próximos aos problemas reais. Assim, devido à esse aspecto e a sua natureza desafiadora, o JSSP tem sido foco de muitos pesquisadores e uma grande variedade de algoritmos tem sido propostos. Esses trabalhos geralmente adotam técnicas de otimização ou de aproximação (JAIN; MEERAN, 1999), também referidas, respectivamente, como algoritmos (ou métodos) exatos e de aproximação (BLAZEWICZ; DOMSCHKE; PESCH, 1996).

1.1 Algoritmos exatos

Os algoritmos exatos buscam, entre todos os escalonamentos possíveis, aqueles que correspondem à solução ótima (ótimo global) do problema, o que geralmente envolve procedimentos de otimização globais computacionalmente caros e dificuldades com relação à convergência. Entre estes, podemos encontrar algoritmos que são principalmente definidos pela eficiência computacional (geralmente referidos como métodos eficientes), tal como os algoritmos de Johnson (1954), Akers (1956) e Jackson (1956), e algoritmos que são principalmente interessados na exploração do espaço das soluções possíveis (geralmente referidos como métodos enumerativos). No primeiro caso, uma solução ótima é construída

seguindo-se a um simples conjunto de regras as quais determinam exatamente a ordem de processamento e, no segundo caso, todas as soluções viáveis são geradas, uma a uma, e procedimentos de eliminação são utilizados no intuito de restringir o espaço de busca. Os principais métodos enumerativos são as técnicas de programação matemática, como a Programação Inteira Mista (do inglês *Mixed Integer Programming*) de Manne (1959), e os vários algoritmos de Ramificação e Poda (do inglês *Branch and Bound Algorithms*) (LENSTRA; RINNOOY KAN, 1973, CARLIER; PINSON, 1989, APPLGATE; COOK, 1991, BRUCKER; JURISCH; SIEVER, 1994, etc.).

Conforme mencionado por Jain e Meeran (1999), os algoritmos exatos não adquiriram ainda o nível requerido para solucionar os problemas de escalonamento da produção em oficina de máquinas, especialmente quando considera-se um grande número de máquinas e serviços. De acordo com os autores, não existem métodos eficientes capazes de resolver problemas onde o número de máquinas e serviços é maior do que três e, no caso dos métodos enumerativos, como o tempo de CPU requerido geralmente aumenta exponencialmente ou como polinômios de alto grau para um aumento linear no tamanho do problema, mesmo o pequeno sucesso obtido pelos algoritmos de Ramificação e Poda deve ser principalmente atribuído à tecnologia disponível, mais do que às técnicas utilizadas. Essa afirmação é sustentada por um estudo recente de Klemmt et al. (2009) que demonstra que a formulação de Manne, que ainda hoje apresenta a melhor performance para solução do JSSP pela Programação Inteira Mista, apresenta bons resultados apenas quando o número de variáveis binárias definido por $m \times n \times (n - 1)$, onde m é o número de máquinas e n o número de serviços, é menor que 1.000 sendo que quando este valor ultrapassa 10.000 o método encontra dificuldades mesmo para encontrar soluções válidas.

1.2 Algoritmos de aproximação

Contrariamente aos algoritmos exatos, os algoritmos de aproximação geram geralmente boas, mas não necessariamente ótimas, soluções em tempos de processamento aceitáveis. Dentre estes procedimentos podemos encontrar uma grande variedade de famílias de algoritmos que vão desde Regras de Ordem de Despacho (do inglês *Priority Dispatch Rules*) (PANWALKAR; ISKANDER, 1977) e Heurísticas Baseadas no Ponto de Gargalo (do

inglês *Bottleneck Based Heuristics*) (ADAMS; BALAS; ZAWACK, 1988, DEMIRKOL et al., 1997; WENQI; AIHUA, 2004, ZHANG ; LI ; LI, 2008) até métodos de inteligência artificial, como as Técnicas de Satisfação de Restrições (do inglês *Constraint Satisfaction Techniques*) (DORNDORF; PESCH; PHAN-HUY, 2000, FROMHERZ, 2001), Redes Neurais (do inglês *Neural Networks*) (WILLEMS; ROODA, 1994, FOO; TAKEFUJI; SZU, 1995, SABUNCUOGLU; GURGUN, 1996) e Otimização por Colônia de Formigas (do inglês *Ant Colony Optimization Algorithms*) (COLORNI et al., 1993, HEINONEN; PETTERSSON, 2007, FIGLALI et al., 2009), contudo seu relativo sucesso é principalmente atribuído a classe de procedimentos de aproximação denominada Heurísticas de Busca Local que inclui as famosas famílias de algoritmos conhecidas como Recozimento Simulado (do inglês *Simulated Annealing*) (VAN LAARHOOVEN; AARTS; LENSTRA, 1992, AZIZI ; ZOLFAGHARI, 2004), Algoritmos Genéticos (do inglês *Genetic Algorithms*) (ONO ; YAMAMURA ; KOBAYASHI, 1996, SHI, 1997, CHENG; GEN; TSUJIMURA, 1996, 1999), Busca Tabu (do inglês *Taboo Search*) (DELL'AMICO; TRUBIAN, 1993, TAILLARD, 1994, NOWICKI; SMUTNICKI, 1996, ZHANG et al., 2007) e, mais recentemente, Otimização por Multidão de Partículas (do inglês *Particle Swarm Optimization*) (LIAN; GU; JIAO, 2006) e, até agora, nenhum algoritmo desenvolvido é capaz de encontrar soluções ótimas em tempos de processamento praticáveis para todos os problemas testes propostos na literatura, de forma que investigações estão ainda em progresso no intuito de melhorar suas performances. Uma extensiva discussão sobre as famílias de algoritmos aplicadas na solução do problema de escalonamento da produção em oficinas de máquinas é apresentada por Błazewicz, Domschke e Pesch (1996) e por Jain e Meeran (1998, 1999).

1.2.1 Heurísticas de Busca Local

As Heurísticas de Busca Local, também referidas como técnicas de geração de vizinhança, são caracterizadas pelo fato de que, a cada iteração, um grupo de novas soluções (definido como vizinhança) é gerado na vizinhança de um conjunto de soluções “genitoras” introduzindo-se pequenas perturbações em cada uma das soluções disponíveis. Conforme representado na Fig. 1, estas perturbações são geradas através da aplicação de operadores que são aqui definidos como operadores de geração de vizinhança. A cada passo, as novas soluções genitoras são formadas por um procedimento de seleção entre as perturbações

geradas. Tal procedimento elimina uma parte das soluções no intuito de obter um novo grupo de soluções genitoras, o qual será utilizado para a geração da vizinhança na próxima iteração. O processo continua até que um dado critério de parada seja satisfeito. De acordo com a escolha da técnica de geração de vizinhança e do processo de seleção, diferentes heurísticas são obtidas. Adicionalmente, a escolha específica do método que irá gerar as soluções genitoras iniciais, do operador que será aplicado para geração da vizinhança, e dos critérios de seleção e de parada, permite que seja gerada uma grande variedade de algoritmos, formando numerosas famílias de algoritmos para cada heurística. Estes algoritmos podem ser determinísticos ou não-determinísticos de acordo com a aplicação de procedimentos randômicos¹: quando nenhum procedimento randômico é utilizado na geração de vizinhança ou na seleção das novas soluções genitoras, o algoritmo é determinístico, quando procedimentos randômicos são utilizados, ele torna-se não determinístico.

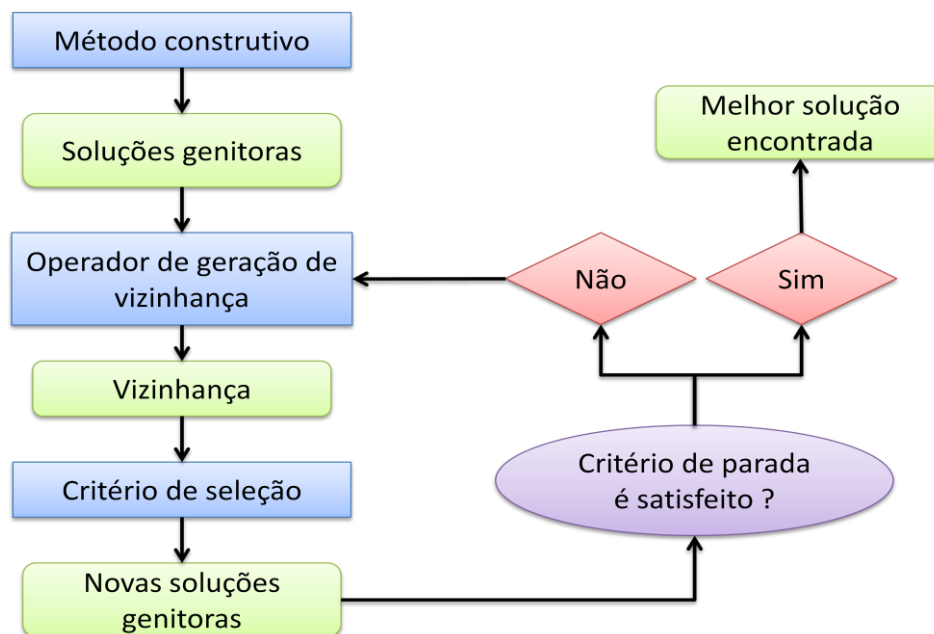


Figura 1 – Esquema das Heurísticas de Busca Local.

Fonte: O autor, 2010.

Atualmente a heurística Busca Tabu, desenvolvida por Glover (1989, 1990), é considerada como uma das mais eficazes para solução do problema de escalonamento da produção em oficina de máquinas. Nesta heurística a cada iteração uma única solução genitora da origem a vizinhança e a nova solução genitora é formada pela melhor solução

¹Gerados segundo uma distribuição uniforme.

dentro da vizinhança não pertencente a lista tabu – a lista tabu é uma lista de soluções proibidas que é aplicada no intuito de tentar prevenir que o processo de busca fique “empacado” em um ótimo local. Sua primeira aplicação em problemas de escalonamento da produção em oficina de máquinas é atribuída a Taillard (1994). Desde então foram sugeridas numerosas melhoras para o algoritmo de Taillard, e dentre as mais importantes contribuições estão os trabalhos desenvolvidos por Dell’Amico e Trubian (1993), Barnes e Chambers (1995), Chambers e Barnes (1996), Nowicki e Smutnicki (1996), e mais recentemente por Zhang et al. (2007).

O Recozimento Simulado é uma técnica de busca randômica que foi inicialmente introduzida como uma analogia à um algoritmo utilizado na física estatística para a simulação computacional do processo de recozimento de um sólido para seu estado fundamental, ou seja, com energia mínima. Em contraste com o Busca Tabu, esta heurística não determinística tenta escapar dos mínimos locais aplicando uma função de probabilidade de transição geralmente definida pelo critério de Metropolis (METROPOLIS et al., 1953). O processo iterativo do Recozimento Simulado pode ser explicado da seguinte forma: uma solução vizinha é gerada randomicamente e comparada com a sua solução genitora. Se a solução gerada melhora a qualidade da solução atual ela é aceita e tida como nova solução genitora. Se não, a função de probabilidade de transição é utilizada para determinar se a solução candidata deverá ser aceita ou rejeitada. Se o valor da probabilidade de transição for maior do que um valor gerado randomicamente, então a nova solução gerada, apesar de pior do que a sua solução genitora, é aceita. Se a transição desta solução é rejeitada, outra solução vizinha será escolhida e analisada. Os algoritmos desenvolvidos por Matsuo, Suh e Sullivan (1988) e Van Laarhoven, Aarts e Lenstra (1992) representam atualmente as formas convencionais de aplicação do Recozimento Simulado ao JSSP sendo que o algoritmo de Van Laarhoven, que introduziu o método de geração de vizinhança atualmente conhecido como N1, é mais referenciado na literatura por ter apresentado melhores resultados. Nesta forma convencional a busca começa com uma alta probabilidade de transição permitindo que mesmo as piores soluções sejam aceitas. Contudo, a medida em que a busca continua, a chance de que as soluções com pouca qualidade sejam aceitas é reduzida. Na última década foram sugeridas algumas adaptações para os modelos convencionais. Stenhöfel, Albrecht e Wong (2003) observaram em seu estudo computacional com problemas testes apresentados na literatura, que a maioria dos resultados ótimos e próximos aos ótimos apresentam um número reduzido de caminhos críticos e, com base nesta observação, desenvolveram uma adaptação

do algoritmo de Van Laarhoven et al. onde a vizinhança é gerada dando preferência aos movimentos que proporcionam uma maior redução do número desta variável. Azizi e Zolfaghari (2004) propuseram um algoritmo chamado Recozimento Simulado Adaptativo, onde a mudança de temperatura é baseada no número consecutivo de movimentos que melhoram a solução encontrada, permitindo assim que o algoritmo fuja dos mínimos locais mesmo após um grande número de iterações. Conforme observado por Jain e Meeran (1998) e verificado no estudo bibliográfico realizado, apesar das grandes contribuições realizadas pelos diversos autores, a aplicação de algoritmos gerados unicamente com base na heurística Recozimento Simulado para o problema de escalonamento da produção em oficina de máquinas, permanece ainda muito pobre e não leva a bons resultados, contudo suas idéias básicas - como as técnicas de geração de vizinhança e o critério de Metropolis - podem ser facilmente introduzidas em outras aproximações de busca local. Como exemplo, temos o algoritmo Colisão de Partículas (do inglês *Particle Collision*), até então aplicado à problemas contínuos (ver SACCO; OLIVEIRA, 2005, SACCO; OLIVEIRA; PEREIRA, 2006, KNUPP; SILVA NETO; SACCO, 2009), onde o critério de Metropolis é utilizado para determinar se o algoritmo deve ou não recomeçar a partir de uma nova solução inicial randômica.

Em contraste com Busca Tabu e o Recozimento Simulado, os Algoritmos Genéticos e o Otimização por Multidão de Partículas (neste texto algumas vezes referido apenas como Multidão de Partículas) são heurísticas populacionais (ou seja, o conjunto de soluções genitoras contém vários elementos), as quais foram inspiradas em eventos observados na natureza. A heurística Algoritmos Genéticos foi elaborada com base em modelos abstratos da evolução natural onde a qualidade dos “indivíduos” (soluções) constrói um maior nível de compatibilidade com o ambiente (restrições do problema). Nesta heurística, a cada iteração uma vizinhança é formada por operações de cruzamento entre pares de soluções genitoras que são selecionados de acordo com uma função de probabilidade. Esta probabilidade é definida em função do índice de eficiência de cada solução genitora, de forma a permitir que soluções melhores tenham maior probabilidade de serem selecionadas. Adicionalmente, também são aplicados sobre alguns dos vizinhos gerados, operadores de mutação no intuito de forçar o algoritmo a vasculhar novos espaços de busca. Conforme mencionado por Jain e Meeran (1999), apesar do fato de muitos elaborados algoritmos terem sido propostos, a construção de convenientes representações do espaço de soluções potenciais para o problema de escalonamento, coerente com as operações de cruzamento e mutação, é ainda considerada como um difícil problema. Adicionalmente, muitos dos algoritmos construídos com base

nesta heurística são incapazes de convergir para soluções ótimas. Já no caso do Multidão de Partículas, o comportamento social de bandos de pombos e peixes são a fonte de inspiração de seus conceitos. Nesta heurística a vizinhança é formada pelo cruzamento de cada solução genitora com a melhor solução entre todas as soluções geradas e a melhor solução visitada por cada partícula. Lian, Gu e Jiao (2006) demonstraram que, quando aplicando alguns dos operadores de cruzamento e mutação dos Algoritmos Genéticos no Otimização por Multidão de Partículas e comparando as duas heurísticas, a segunda leva a melhores resultados e apresenta uma convergência nitidamente mais rápida. Busca Tabu, Algoritmos Genéticos e Multidão de Partículas são geralmente da família de heurísticas não-determinísticas mas, do ponto de vista abstrato, suas estruturas podem ou não envolver passos randômicos de acordo com suas implementações.

1.2.2 Hibridização de algoritmos

Conforme anteriormente citado, apesar do sucesso parcial atingido pelos pesquisadores no desenvolvimento de poderosas técnicas, até agora não foi apresentado nenhum algoritmo capaz de encontrar soluções ótimas, em tempos de processamento praticáveis, para todos os problemas testes propostos na literatura. Neste contexto, uma possível aproximação consiste no uso de algoritmos híbridos que, em analogia às misturas químicas, podem ser classificados² como: homogêneos, quando gerados pela inserção de procedimentos específicos de uma família de algoritmos, como a lista tabu do Busca Tabu ou o critério de Metropolis do Recozimento Simulado, em algoritmos pertencentes a outras famílias de naturezas diferente, gerando novos algoritmos com características próprias; heterogêneos quando gerados pelo uso simultâneo de algoritmos de naturezas diferentes, i.e. determinísticos e não-determinísticos ou individuais (com apenas uma solução genitora) e populacionais, de forma que cada algoritmo conserva as suas próprias características; ou mistos, quando os novos algoritmos gerados apresentam uma parte homogênea e outra heterogênea, e este tem sido o foco de diversos autores. Pezzella e Merelli (2000), baseados na idéia de que os resultados gerados pelo Busca Tabu são fortemente dependentes da solução inicial, apresentaram um algoritmo híbrido heterogêneo onde o método de Deslocamento do Ponto de Gargalo de Adams, Balas e Zawack (1988) é utilizado para gerar estas soluções

² Esta classificação é sugerida pelo autor da presente tese.

iniciais. Wang e Zheng (2001) propuseram um algoritmo híbrido heterogêneo onde inicialmente um Algoritmo Genético é utilizado para gerar o conjunto de soluções iniciais, posteriormente o Recozimento Simulado, a cada variação de temperatura, aplica o critério de Metropolis a cada solução até que seja atingida a condição de equilíbrio, e então o Algoritmo Genético utiliza as soluções encontradas pelo Recozimento Simulado para continuar a evolução paralela de todas as soluções. Azizi e Zolfaghari (2004) desenvolveram um algoritmo híbrido homogêneo, adicionando uma lista tabu ao seu algoritmo Recozimento Simulado Adaptativo, o que melhorou significativamente os resultados encontrados. Mais recentemente, Zhang et al. (2008) apresentaram um algoritmo híbrido heterogêneo no qual inicialmente a heurística Recozimento Simulado é utilizada para encontrar as boas soluções dentro do “big valley” (região onde se encontram as soluções ótimas e próximas às soluções ótimas) e posteriormente a heurística Busca Tabu é aplicada no intuito de intensificar o processo de busca.

1.3 Objetivos específicos desta tese

Esta tese tem como principal objetivo propor e analisar novos modelos de hibridização para solução do problema de escalonamento de produção em oficina de máquinas, que possam ser aplicados com os diferentes algoritmos já desenvolvidos, buscando gerar novos algoritmos híbridos mais robustos e que beneficiem da qualidade de seus algoritmos de base, superando as suas deficiências.

Com base neste objetivo, em um primeiro momento, são apresentados neste trabalho três modelos de hibridização desenvolvidos com base nas idéias de hibridização propostas por Souza de Cursi (2009). Estes modelos são analisados com dois algoritmos de busca local, cuja hibridização, de acordo com a pesquisa efetuada, ainda não fora analisada, que são respectivamente, o algoritmo Busca Tabu apresentado por Nowicki e Smutnicki (1996), com uma versão determinística da sua estrutura de vizinhança e sem retro-propagação, e o algoritmo Multidão de Partículas Similar conforme proposto por Lian, Gu e Jiao (2006), mostrando quais aspectos devem ser considerados no intuito de se obter um algoritmo melhorado, mais robusto e capaz de obter melhores resultados do que cada um dos algoritmos separados, com um custo computacional razoável. Em um segundo momento, é proposto

nesta tese como um quarto modelo de hibridização uma adaptação do método Colisão de Partículas (até então aplicado apenas à solução de problemas contínuos) para o problema de escalonamento da produção em oficina de máquinas, envolvendo o algoritmo Busca Tabu anteriormente citado, que é utilizado como operador de exploração local, e a técnica de mutação M7 apresentada por Lian, Gu e Jiao (2006), que é utilizada como operador de perturbação.

O restante desta tese é organizada da seguinte maneira: No capítulo 2 o problema de escalonamento da produção em oficina de máquinas é introduzido. O capítulo 3 trata da solução do problema e é distribuído da seguinte forma: As seções 3.1 e 3.2 apresentam respectivamente uma revisão das heurísticas Busca Tabu e Optimização por Multidão de Partículas e introduz os algoritmos cuja hibridização será aqui analisada. As idéias de hibridização concebidas por Souza de Cursi (2009), assim como os três modelos desenvolvidos com base nestas idéias, são apresentados na seção 3.3. Na seção 3.4, o método Colisão de Partículas é apresentado juntamente com a adaptação desenvolvida para o problema de escalonamento da produção em oficina de máquinas, a qual é definida como um quarto modelo de hibridização. O capítulo 4 apresenta e discute os resultados obtidos pelas diferentes possibilidades de hibridização. Finalmente, o capítulo 5 sumariza as contribuições desta tese e apresenta suas conclusões e algumas proposições para trabalhos futuros.

2 PROBLEMA DE ESCALONAMENTO DA PRODUÇÃO EM OFICINA DE MÁQUINAS

Conforme descrito por Błazewicz, Domschcke e Pesch (1996), uma oficina de máquinas (do inglês *job shop*) é formada por um grupo de máquinas distintas que são capazes de realizar todas as operações necessárias ao processamento de diferentes serviços. Neste ambiente, cada serviço visita cada máquina uma única vez, sendo que a ordem das máquinas em que cada serviço é processado é previamente definida e pode variar de um serviço para outro. Dessa forma, um serviço pode ser definido como uma sequência de operações onde cada operação representa a parte do serviço que deve ser processada em uma determinada máquina durante um tempo de processamento préconhecido e fixo. Adicionalmente, cada serviço deve ser processado por apenas uma máquina de cada vez, o que introduz uma restrição de precedência entre as operações de um mesmo serviço (restrição de serviço), ou seja, o momento inicial de processamento de cada operação deve ser sempre maior do que o momento final de processamento da sua operação precedente na ordem de serviço sendo que, uma vez que o processamento de uma operação começa, o mesmo não pode ser interrompido (restrição de preempção). De forma similar, cada máquina pode processar apenas um serviço por vez (restrição de máquina) e, uma vez que a sequência de máquinas de cada serviço é fixa, quando considerando um conjunto de serviços que devem ser simultaneamente processados, o problema consiste em encontrar uma sequência de serviços para cada máquina de forma que o conjunto de sequências definido (escalonamento) respeite às restrições acima descritas e minimize o tempo total de processamento de todos os serviços, ou seja, o tempo final de processamento da última operação completada (no inglês referido como *makespan*).

Para melhor entendimento do problema, considere uma confecção de roupas masculinas cuja oficina de máquinas contém 6 máquinas distintas (M1, M2, M3, M4, M5, M6) capazes de realizar as operações necessárias ao processamento de seis diferentes modelos de roupas (por exemplo: camisa, camiseta, calça, suéter, short e casaco), respectivamente definidos como serviços S1, S2, S3, S4, S5 e S6. A ordem de processamento de cada serviço através das máquinas assim como o tempo de processamento de cada serviço em cada máquina são definidos conforme Tab.1, a seguir. Já uma solução³ para o problema pode ser

³Na literatura um escalonamento, definido como o conjunto das sequências específicas de serviços em cada máquina, é também referido como solução. O leitor deve ter em mente que esta expressão não concerne à solução do problema de otimização a qual é referida na literatura como solução ótima. Nesta tese todas as expressões: solução, possível solução e

determinada, definindo-se para cada máquina uma sequência de serviços, conforme Fig. 2.

Tabela 1 – Ordem das máquinas e tempo de processamento de cada serviço em cada máquina para o problema da confecção de roupas masculinas. M = máquina; UT = unidade de tempo. Os dados desta tabela foram gerados com base no problema FT6 (FISHER; THOMPSON, 1963).

Serviços	1ª máquina		2ª máquina		3ª máquina		4ª máquina		5ª máquina		6ª máquina	
	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT
S1	3	1	1	3	2	6	4	7	6	3	5	6
S2	2	8	3	5	5	10	6	10	1	10	4	4
S3	3	5	4	4	6	8	1	9	2	1	5	7
S4	2	5	1	5	3	5	4	3	5	8	6	9
S5	3	9	2	3	5	5	6	4	1	3	4	1
S6	2	3	4	3	6	9	1	10	5	4	3	1

Fonte: O autor, 2010.

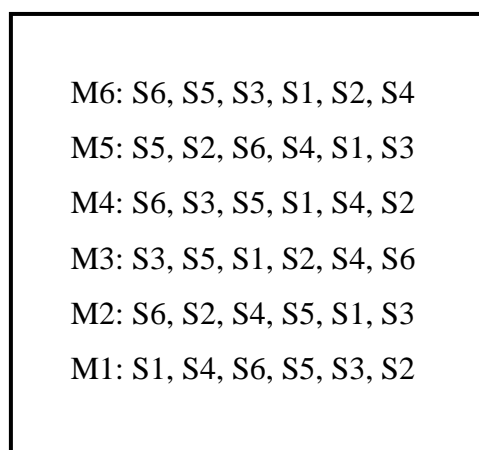


Figura 2 – Solução para o problema da confecção de roupas masculinas representada pelo código de sequência de serviços por máquinas.

Fonte: O autor, 2010.

Com base nos dados apresentados na Tab. 1, esta solução pode ser também representada em um gráfico de Gantt, conforme Fig. 3. Nessa figura, o sequenciamento (ou solução) definido a priori é representado em um gráfico de Gantt, onde cada linha representa uma máquina, cada cor representa um diferente serviço (amarelo para S1, azul para S2, rosa para S3, verde para S4, cor de pele para S5, e laranja para S6), e cada quadrado representa uma operação e o espaço de tempo em que a mesma deve ser processada. As operação que estão com contorno em preto representam o caminho crítico, ou seja, a sequência de operações que define o tempo total de processamento do grupo de serviços, na literatura referido como *makespan*. Neste gráfico pode-se verificar que o *makespan* correspondente a solução é de 71 UT (unidades de tempo). Também é possível observar que existe uma grande

quantidade de espaços em branco entre as operações o que indica que há um longo tempo ocioso nas máquinas. Novas soluções podem ser geradas apenas mudando-se a sequência de serviços nas máquinas. Temos ao total $(n!)^m$ soluções diferentes (onde n é o número de serviços e m o número de máquinas) sendo que, de acordo com as restrições do problema, nem todas são soluções viáveis.

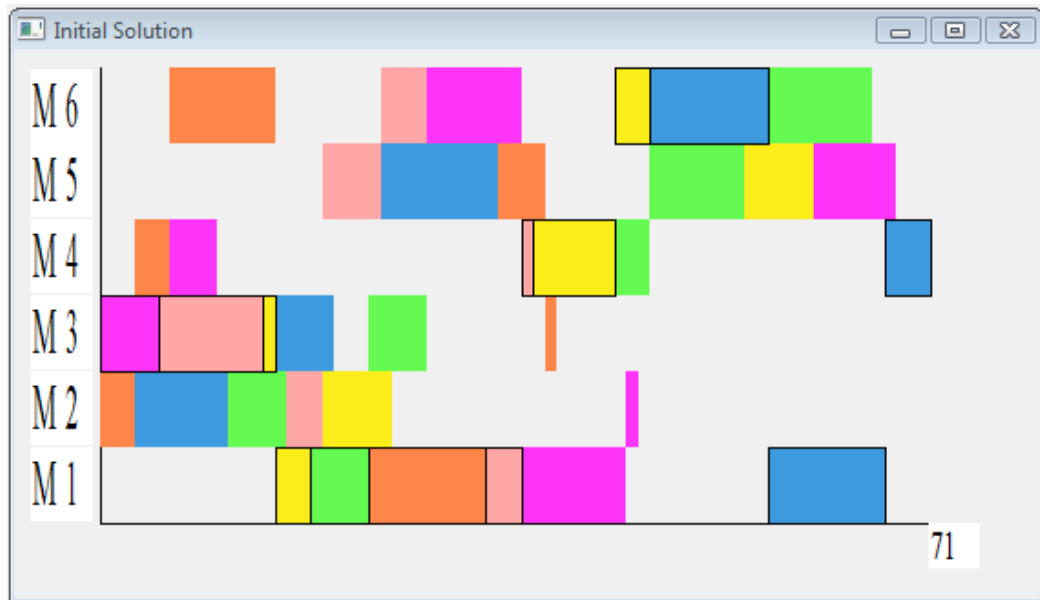


Figura 3 – Solução para o problema da confecção de roupas masculinas representada em um gráfico de Gantt.

Fonte: O autor, 2010.



Figura 4 – Solução ótima em Gantt para o problema da confecção de roupas masculinas.

Fonte: O autor, 2010.

A Fig. 4, apresenta uma das possíveis soluções ótimas para este problema (*makespan* = 55 UT). Conforme pode ser verificado quando comparando as Figs. 3 e 4, uma solução ótima pode gerar uma grande redução no tempo total de produção do grupo de serviços e, conseqüentemente, também uma grande redução no tempo ocioso de máquina, o que pode ser refletido em uma redução significativa dos custos de produção, tornando a empresa mais competitiva.

2.1 Modelagem matemática

Existe na literatura uma série de formulações para o problema de escalonamento da produção em oficina de máquinas as quais foram introduzidas de forma a gerar base para aplicação dos diferentes métodos de solução propostos. Nas próximas secções algumas dessas modelagens são apresentadas.

2.1.1 Modelagem clássica

Conforme apresentado por Jain e Meeran (1999), o JSSP (referido como Π_j) consiste de um conjunto finito J de n serviços, $\{J_i\}_{i=1}^n$, que devem ser processados em um conjunto finito M de m máquinas, $\{M_k\}_{k=1}^m$. Cada serviço J_i deve ser processado em todas as máquinas e consiste de uma cadeia ou complexo de m_i operações, $O_{i1}, O_{i2}, \dots, O_{im_i}$, que devem ser processadas em uma determinada ordem (restrição de precedência). Há no total N operações, onde $N = \sum_{i=1}^n m_i$. O_{ik} é a operação do serviço J_i que deve ser processada pela máquina M_k por um período de tempo de processamento τ_{ik} fixo e, uma vez que o processamento de uma operação começa, o mesmo não pode ser interrompido (restrição de preempção). Cada serviço possui a sua ordem de processamento individual através das máquinas a qual é independente dos outros serviços. Cada serviço pode ser processado por apenas uma máquina e cada máquina pode processar apenas um serviço por vez (restrição de capacidade). O momento no qual todas as operações de todos os serviços são completadas é referido como tempo total de processamento, C_{max} (no inglês referido como *makespan*). O único objetivo considerado neste problema é o de determinar momentos iniciais de processamento para cada operação,

$t_{ik} \geq 0$, de forma a minimizar o tempo total de processamento, satisfazendo todas as restrições de precedência e capacidade (no intuito de gerar apenas escalonamentos viáveis).

$$C_{max}^* = \min(C_{max}) = \min_{\text{escalonamentos viáveis}} (\max(t_{ik} + \tau_{ik}) : \forall J_i \in J, M_k \in M) \quad (1)$$

A dimensão de cada problema Π_j é definida como $n \times m$ e N é frequentemente assumido como $n \times m$, uma vez que $m_i = m$ para cada serviço $J_i \in J$ e que cada serviço deve ser processado exatamente uma vez em cada máquina. Neste caso o número total de soluções é dado por $(n!)^m$ e não é difícil verificar que, para grandes JSSP, uma enumeração completa de todas essas possibilidades para identificar os escalonamentos viáveis e os ótimos não é nada prática, por exemplo, um problema de dimensão 20×10 possui $7,2651 \times 10^{13}$ soluções possíveis o que excede a idade postulada da terra em microssegundos.

2.1.2 Modelagem disjuntiva

Considerando que $m_i = m$, $i = 1, 2, \dots, n$, a ordem de processamento de cada serviço J_i através das máquinas pode ser representada por uma dada permutação $\phi_i = (\phi_1^i, \phi_2^i, \dots, \phi_m^i)$, a qual indica que o serviço J_i deve ser processado primeiro na máquina ϕ_1^i , depois na máquina ϕ_2^i , e assim sucessivamente. Desta forma, pode-se dizer que, para que o vetor $(t_{ik} : \forall i = 1, 2, \dots, n, k = 1, 2, \dots, m)$ represente um escalonamento viável, ele deve estar sujeito às seguintes restrições (APPLEGATE; COOK, 1991):

$$t_{ik} \geq 0 \quad \forall i = 1, 2, \dots, n, \quad k = 1, 2, \dots, m \quad (2)$$

$$t_{i\phi_k^i} \geq t_{i\phi_{k-1}^i} + \tau_{i\phi_{k-1}^i} \quad \forall i = 1, 2, \dots, n, \quad k = 1, 2, \dots, m \quad (3)$$

$$t_{jk} \geq t_{ik} + \tau_{ik} \text{ ou } t_{ik} \geq t_{jk} + \tau_{jk} \quad \forall i, j = 1, 2, \dots, n, \quad k = 1, 2, \dots, m \quad (4)$$

onde: a desigualdade (2) força os tempos iniciais de processamento de cada operação a serem positivos; a desigualdade (3) diz que apenas uma operação pode ser processada em cada

máquina por vez; e ambas as desigualdades em (4) impõem a restrição de precedência entre as operações do mesmo serviço. Para incorporar a função objetivo, o tempo total de processamento, C_{max} , deve estar sujeito à seguinte restrição:

$$C_{max} \geq t_{i\phi_m^i} + \tau_{i\phi_m^i} \quad \forall i = 1, 2, \dots, n \quad (5)$$

E o JSSP pode ser então reescrito como um problema de programação disjuntiva da seguinte forma:

$$\text{minimize}\{C_{max} \text{ sujeito a (2), (3), (4) e (5)}\} \quad (6)$$

2.1.3 Modelagem Inteira Mista

Segundo Applegate e Cook (1991) o problema de programação disjuntiva pode ser transformado em um problema de programação inteira mista através da introdução de uma variável binária Y_{ij}^k para cada uma das condições (4) e impondo a nova restrição:

$$\begin{aligned} t_{ik} &\geq t_{jk} + \tau_{jk} - K \times Y_{ij}^k & t_{jk} &\geq t_{ik} + \tau_{ik} - K \times (1 - Y_{ij}^k) \\ \forall i, j &= 1, 2, \dots, n, & k &= 1, 2, \dots, m \end{aligned} \quad (7)$$

onde K é uma constante com um grande valor. A interpretação é que Y_{ij}^k é igual a 1 se J_i é processado antes de J_j na máquina M_k e 0 no caso contrário.

2.2 Problemas padrões apresentados na literatura

Na literatura pesquisada foram encontrados uma série de problemas padrões que são

utilizados para teste dos algoritmos desenvolvidos para solução do problema de escalonamento da produção em oficina de máquinas (ver FISHER; THOMPSON, 1963, LAWRENCE, 1984, ADAMS; BALAS; ZAWACK, 1988, APPELATE; COOK, 1991, STORER; WU; VACCARI, 1992, e YAMADA; NAKANO, 1992, 1997). Estes problemas são definidos basicamente, pelo número de máquinas e serviços considerados, pela ordem de máquinas em que cada serviço deve ser processado e pelo tempo de processamento de cada serviço em cada máquina. A complexidade dos diferentes problemas está diretamente relacionada ao seu número de máquinas e serviços, contudo a forma de definição das outras variáveis, em especial a escolha da sequência de máquinas em que cada serviço deve ser processado, também gera uma influência direta em seu comportamento, de forma que diferentes problemas com mesmo número de máquinas e serviços, podem apresentar maior ou menor complexidade de acordo com o algoritmo aplicado em sua solução. Na presente tese foram utilizados para análise dos modelos de hibridização propostos os problemas padrões FT6 (com 6 máquinas e 6 serviços), FT10 (com 10 máquinas e 10 serviços) e FT20 (com 5 máquinas e 20 serviços) propostos por Fisher e Thompson (1963) e os problemas ABZ5 e ABZ6 (com 10 máquinas e 10 serviços), e ABZ7, ABZ8 e ABZ9 (com 15 máquinas e 20 serviços) propostos por Adams, Balas e Zawack (1988). Estes problema são apresentados no anexo 1 desta tese.

3 SOLUÇÃO DO PROBLEMA DE ESCALONAMENTO DA PRODUÇÃO EM OFICINA DE MÁQUINAS

Conforme anteriormente informado, para solução do problema de escalonamento da produção em oficina de máquinas são propostos nesta tese quatro modelos de hibridização, sendo três deles baseados nas idéias de hibridização propostas por Souza De Cursi (2009) e o quarto baseado no método Colisão de Partículas. Os três primeiros modelos são analisados com dois algoritmos de busca local, que são respectivamente, o algoritmo Busca Tabu apresentado por Nowicki e Smutinicki (1996), com uma versão determinística da sua estrutura de vizinhança e sem retro-propagação, e o algoritmo Otimização por Multidão de Partículas Similar conforme proposto por Lian, Gu e Jiao (2006), enquanto que o quarto modelo, é analisado com o algoritmo Busca Tabu anteriormente citado, que é utilizado como operador de exploração local, e a técnica de mutação M7 apresentada por Lian Gu e Jiao (2006), que é utilizada como operador de perturbação. Nas próximas seções as heurísticas Busca Tabu, Multidão de Partículas e Colisão de Partículas são revistas e modelos de hibridização propostos nesta tese são introduzidos.

3.1 Busca Tabu

A heurística Busca Tabu começa com uma única solução inicial, gerada randomicamente ou através de algum método construtivo, a qual é armazenada como solução atual e como melhor solução até então encontrada. A cada iteração uma vizinhança (grupo de novas soluções) é gerada, pela aplicação de pequenas perturbações (geralmente denominadas movimentos) na solução atual e então esta solução é substituída pelo melhor vizinho não tabu, ou seja, pela melhor solução dentro da vizinhança cujo movimento pelo qual ela é gerada não pertence a lista tabu. A lista tabu é um vetor contendo os movimentos inversos aos utilizados na geração das t últimas soluções atuais, e é introduzida como uma forma de prevenir que o processo de busca fique empacado em um ótimo local. Critérios de aspiração podem ser definidos no intuito de permitir a substituição da solução atual por uma solução cujo movimento gerador pertence à lista tabu caso seja bom para o processo de busca como, por exemplo, quando a solução é melhor do que a melhor solução já encontrada. A melhor

solução é substituída pelo melhor vizinho, caso este último apresente um menor *makespan* do que o do primeiro. O processo continua até que um dado critério de parada seja satisfeito.

Gere uma solução inicial.

Guarde esta solução como solução atual e melhor solução.

Enquanto critério de parada não é satisfeito

 Gere uma vizinhança.

 Substitua a solução atual pelo melhor vizinho “não tabu”.

Se o *makespan* do melhor vizinho é menor do que o *makespan* da melhor solução
 Substitua a melhor solução pelo melhor vizinho.

Fim Se

Atualize a lista tabu adicionando o movimento inverso ao movimento gerador da nova solução atual e removendo o último movimento da lista (caso o tamanho da lista tabu seja maior que t).

Fim Enquanto

Retorne a melhor solução encontrada.

Figura 5 – Algoritmo para a heurística Busca Tabu.

Fonte: O autor, 2010.

A Fig. 5 apresenta um algoritmo geral para a heurística Busca Tabu. Nesse algoritmo o processo de geração de vizinhança tem um impacto direto na eficiência do método e várias estruturas de vizinhança tem sido apresentadas na literatura como, por exemplo, as vizinhanças geradas pela troca de operações no caminho crítico definidas como N1, N2, N3, N4, N5 e N6 (ver ZHANG et al., 2007). Entre elas a estrutura de vizinhança desenvolvida por Nowicki e Smutinicki (1996), denominada N5, foi a que introduziu o verdadeiro avanço tanto em termos de eficiência quanto de eficácia para o problema de escalonamento da produção em oficina de máquinas. Esta estrutura gera uma vizinhança substancialmente menor do que as outras e será utilizada neste trabalho.

3.1.1 Método de geração de vizinhança N5

Dada uma solução qualquer, um caminho crítico pode ser definido como uma sequência de operações onde:

- a última operação na ordem desta sequência é a operação que é finalizada por último e que, portanto, tem tempo final de processamento igual ao *makespan*;
- a operação precedente a cada operação, exceto à primeira, na ordem desta sequência é a operação que a precede na ordem de máquina ou na ordem de serviço, sendo entre as duas àquela que apresentar maior tempo final de processamento.
- o momento inicial de processamento da primeira operação na ordem desta sequência é igual a zero.

Nas Figs. 3 e 4, as operações com contorno em preto representam possíveis caminhos críticos referentes as respectivas soluções. É importante observar que uma solução pode conter mais do que um caminho crítico, como no caso em que há duas ou mais operações que são finalizadas por último, ou quando as operações que precedem uma dada operação nas ordens de máquina e de serviço são finalizadas em um mesmo momento. Em seu artigo, Nowicki e Smutinicki (1996) afirmam que a escolha do caminho crítico não gera forte influência no resultado final e os autores sugerem que o mesmo seja escolhido de forma randômica. Neste trabalho, como deseja-se que o método Busca Tabu apresente-se como um método determinístico, o caminho crítico é selecionado de acordo com as seguintes prioridades:

- caso duas operações sejam finalizadas por último, a operação escolhida será aquela pertencente a máquina de menor índice;
- caso as operações que precedem uma dada operação nas ordens de máquina e de serviço sejam finalizadas em um mesmo momento, a operação escolhida será aquela precede na ordem de serviço.

O caminho crítico também pode ser subdividido em blocos de operações onde cada bloco é composto pela subsequência de operações sucessivas que devem ser processadas por uma mesma máquina. Pelo método de geração de vizinhança N5, uma vizinhança é formada pela troca entre as duas primeiras e as duas últimas operações de cada bloco de operações no caminho crítico, sendo que no caso do primeiro bloco, apenas as duas últimas operações são trocadas e no caso do último bloco apenas as duas primeiras operações são trocadas. Caso algum bloco seja formado por apenas uma operação, nenhuma troca será realizada e, caso o mesmo seja formado por duas operações, será realizada apenas uma troca. Na Fig. 6 são apresentados a solução para o problema da confecção de roupas masculina (Fig. 3) e os 5 vizinhos formados pelo método N5 para esta solução.

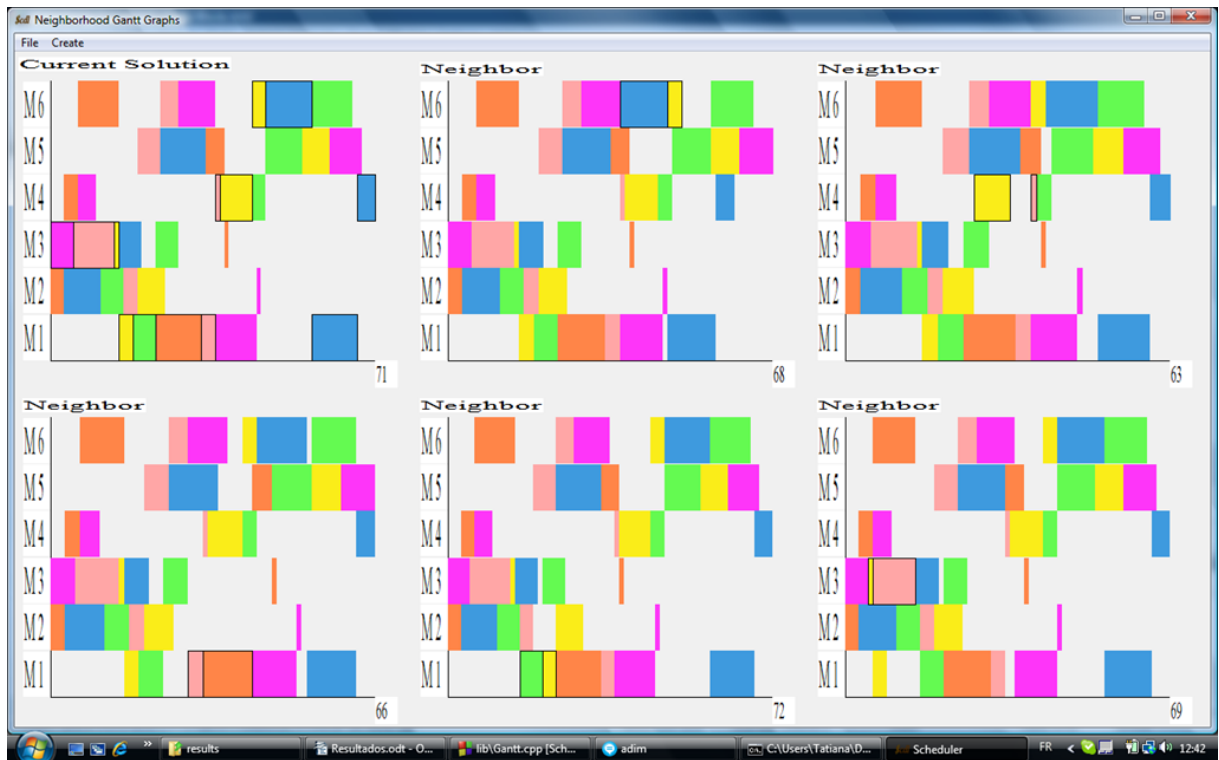


Figura 6 – Vizinhança gerada pelo método N5.

Fonte: O autor, 2010.

O problema do Busca Tabu é que sua performance é muito sensível tanto à solução inicial quanto ao ajuste de seus parâmetros, quando estes últimos não são corretamente definidos de acordo com cada problema específico, o método entra em processos cíclicos impedindo que ótimos globais sejam encontrados. A Tab. 2 apresenta uma análise de sensibilidade para o parâmetro “tamanho da lista tabu” para dez diferentes soluções iniciais geradas randomicamente para o problema FT10 proposto por Fisher e thompson (1963). Os valores marcados em negrito correspondem a melhor solução encontrada para cada uma das soluções apresentadas. Conforme pode ser observado nesta tabela, o Busca Tabu não é um método muito robusto e é incapaz de encontrar soluções ótimas caso os valores exatos para ajuste de seus parâmetros não sejam conhecidos. Uma resposta ótima para este problema (*makespan* = 930 UT) foi encontrada pelo mesmo programa em apenas 15 segundos de tempo de CPU fazendo “tamanho da lista tabu” = 30. O processo cíclico do Busca Tabu pode ser visualizado na Tab. 3, onde é possível observar que, a partir de um determinado valor, mesmo um grande acréscimo para o parâmetro “número máximo de iterações sem melhora do *makespan*” não é capaz de melhorar a solução encontrada. Enquanto o tempo de CPU

aumenta proporcionalmente ao acréscimo desta variável, a solução cai em um ótimo local e outras técnicas são necessárias para melhorar os resultados encontrados.

Tabela 2 – Análise de sensibilidade do *makespan* para o parâmetro “tamanho da lista tabu” para dez soluções iniciais geradas randomicamente para o problema FT10, com os seguintes parâmetros fixos: método de geração de vizinhança = N5; número máximo de iterações sem melhora do *makespan* = 10.000.

Solução	Valor inicial	Tamanho da lista tabu								
		5	6	7	8	9	10	11	12	13
1	1668	1035	971	951	990	949	961	943	949	958
2	1695	951	948	945	948	945	943	954	951	951
3	1977	994	954	972	940	945	938	938	954	934
4	1807	1136	980	963	945	934	953	956	952	955
5	1749	988	1031	940	944	945	946	950	955	946
6	1629	1017	968	1041	949	966	960	951	951	945
7	1922	949	958	954	940	954	963	948	965	942
8	1783	968	956	953	951	951	940	940	947	952
9	1701	1104	969	983	937	949	948	949	954	961
10	1766	1005	1023	962	944	951	948	952	939	957

Fonte: O autor, 2010.

Tabela 3 – Análise de sensibilidade do *makespan* para o parâmetro “número máximo de iterações sem melhora do *makespan*” para dez soluções iniciais geradas randomicamente para o problema FT10, com os seguintes parâmetros fixos: método de geração de vizinhança = N5; tamanho da lista tabu = 5.

Solução	Valor inicial	Número máximo de iterações sem melhora do <i>makespan</i>				
		10	100	1000	10000	100000
1	1668	1128	1056	1035	1035	1035
2	1695	1091	979	951	951	951
3	1977	1350	1088	1021	994	994
4	1807	1162	1136	1136	1136	1136
5	1749	1117	1074	988	988	988
6	1629	1225	1053	1017	1017	1017
7	1922	1044	1017	<u>949</u>	949	949
8	1783	1091	1017	968	968	968
9	1701	1133	1104	1104	1104	1104

Fonte: O autor, 2010.

3.2 Otimização por Multidão de Partículas

O Otimização por Multidão de Partículas é uma heurística baseada em população que foi originalmente desenvolvida por Kennedy e Eberhart (1995) com base no comportamento de cardumes de peixes e bandos de pombos durante a busca pelo alimento. Nesta heurística, cada indivíduo é interpretado como sendo uma partícula i ($1 \geq i \leq NP$) e cada solução é interpretada como sendo a posição atual dessa partícula, a qual é representada por um vetor de dimensão d , onde d é o número de variáveis que devem ser ajustadas. Em suas trajetórias as partículas buscam por uma posição correspondente a solução ótima do problema. A implementação do Otimização por Multidão de Partículas (ver Fig.7) pode ser descrita conforme a seguir: inicialmente as posições iniciais do grupo de partículas são definidas randomicamente. Posteriormente, a cada iteração (k), a posição de cada partícula (x_i) é ajustada de acordo com sua velocidade (v_i), a qual é definida randomicamente por uma solução pertencente ao intervalo de soluções contido entre a melhor solução visitada pela partícula ($pbest_i$) e a melhor solução visitada por todo o grupo ($gbest$).

Gere, para o grupo de partículas, posições iniciais randômicas no espaço d dimensional.

Enquanto critério de parada não é satisfeito

 Calcule a velocidade de cada partícula.

 Calcule a nova posição de cada partícula.

 Calcule o valor correspondente a nova posição de cada partícula de acordo com a função a ser otimizada e, ao mesmo tempo, atualize, caso necessário, $pbest_i$ (de cada partícula) e $gbest$.

Fim Enquanto

Retorne a melhor solução encontrada.

Figura 7 – Algoritmo para o método Otimização por Multidão de Partículas.

Fonte: O autor, 2010.

No caso de problemas com variáveis contínuas, a cada iteração k , v_i e x_i são ajustados conforme as seguintes equações:

$$v_i(k+1) = w \times v_i(k) + c_1 r_1 (pbest_i(k) - x_i(k)) + c_2 r_2 (gbest(k) - x_i(k)) \quad (8)$$

$$x_i(k+1) = x_i(k) + v_i(k+1) \quad (9)$$

O peso de inércia w , inicialmente proposto por Shi e Eberhart (1998), é utilizado para o controle de exploração global e local. Um grande valor para w pode prevenir que as partículas fiquem presas em mínimos locais, e um pequeno valor para w encoraja as partículas a explorarem o mesmo espaço de busca. As constantes c_1 e c_2 são fatores de aprendizagem utilizados para decidir se as partículas se aproximam mais de $pbest$ ou $gbest$. Geralmente, $c_1 = c_2 = 2$. r_1 e r_2 são variáveis randômicas entre 0 e 1.

O Otimização por Multidão de Partículas foi originalmente desenvolvido para solução de problemas com variáveis contínuas. Quando trabalhamos com problemas de otimização combinatória, torna-se necessário definir como as possíveis soluções (ou posições) serão representadas além da forma em que as velocidades e posições serão ajustadas. Lian et al. (2006) propuseram para o problema de escalonamento da produção em oficina de máquinas a heurística Otimização por Multidão de Partículas Similar (do inglês *Similar Particle Swarm Optimization*), onde a posição de cada partícula é representada pelo “código de processamento de serviços” (do inglês *work procedure code*) (ver seção 3.1), o qual considera apenas soluções viáveis, e suas velocidades e posições são ajustadas de acordo com as seguintes equações:

$$v_i(k+1) = pbest_i(k) \theta gbest(k) \quad (10)$$

$$v_i(k+1) = \begin{cases} v_i(k+1) & \text{if } mut_i = 0 \\ M(v_i(k+1)) & \text{if } mut_i = 1 \end{cases} \quad (11)$$

$$x_i(k+1) = x_i(k) \theta v_i(k+1) \quad (12)$$

$$x_i(k+1) = \begin{cases} x_i(k+1) & \text{if } mut_i = 0 \\ M(x_i(k+1)) & \text{if } mut_i = 1 \end{cases} \quad (13)$$

Onde θ e $M(x)$ representam, respectivamente, os operadores de cruzamento e mutação aplicados nos Algoritmos Genéticos e a variável booleana mut_i é um sinal utilizado para indicar se a operação de mutação deve ($mut_i = 1$) ou não ($mut_i = 0$) ser aplicada à partícula i . A cada iteração N partículas são escolhidas randomicamente para sofrer a operação de mutação.

$$\sum_{i=1}^{NP} (mut_i) = N \quad (14)$$

Os autores testaram 4 operadores de cruzamento (C1 - C4) e 10 operadores de mutação (M1 - M10) em três problemas que são utilizados com frequência para teste de diversos métodos apresentados na literatura, definidos como FT6, FT10 e FT20 (FISHER; THOMPSON, 1963), demonstrando que a Otimização por Multidão de Partículas Similar é mais eficiente para a solução de problemas de escalonamento da produção em oficina de máquinas do que os Algoritmos Genéticos (ver Tabs. 4, 5 e 6). Neste trabalho são utilizados o operador de mutação M7 (reposicionamento de serviços), com o reposicionamento de apenas um serviço, e um novo operador de cruzamento definido nessa tese como “C1 com um ponto de corte flutuante”. O código de processamento de serviços e os operadores de cruzamento e mutação aplicados neste trabalho são descritos nas próximas secções.

Tabela 4 – Comparação entre os resultados da Otimização por Multidão de Partículas Similar (OMPS) e dos Algoritmos Genéticos (AG) com a aplicação dos operadores de cruzamento (C1 - C4) e com os operadores de mutação (M1 - M10) para o problema teste FT6. TP = Tamanho da população para a Otimização por Multidão de Partículas Similar e NG = Número de genes para os Algoritmos Genéticos.

Problema	Solução ótima	TP / NG	Mínimo / Máximo / Média em 10 rodadas					
					C1	C2	C3	C4
FT6	55	100 / 20	M1	OMPS	58/60/59.2	59/60/59.5	59/60/59.6	59/60/59.4
				AG	55/59/56.6	55/59/ 56	55/59/56.5	55/59/56.2
			M2	OMPS	57/60/59	59/60/59.5	59/60/59.5	59/60/59.5
				AG	55/59/56.2	55/59/56.3	55/57/56	55/58/ 55.6
			M3	OMPS	59/60/59.3	59/61/59.5	59/60/59.5	59/59/59
				AG	55/58/56.4	55/59/ 56	55/59/56.6	55/58/56.2
			M4	OMPS	59/61/59.3	59/60/59.4	59/60/59.8	59/60/59.4
				AG	55/58/55.9	55/59/55.8	55/58/55.6	55/58/ 55.3
			M5	OMPS	59/60/59.3	59/60/59.5	58/60/59.2	59/61/59.7
				AG	55/59/56	55/59/56.8	55/58/ 55.7	55/59/56.5
			M6	OMPS	59/60/59.2	58/60/59.2	59/60/59.3	59/60/59.1
				AG	55/59/56.4	55/59/56.7	55/58/56.6	55/59/ 55.9
			M7	OMPS	58/60/59.1	59/61/59.5	58/60/59.4	59/60/59.3
				AG	55/59/56	55/58/ 55.8	55/59/56.4	55/58/56.7
			M8	OMPS	58/60/59.2	59/60/59.4	59/60/59.5	59/61/59.7
				AG	55/59/56.2	55/59/ 55.7	55/58/55.9	55/58/55.8
			M9	OMPS	59/60/59.4	58/60/59.1	59/60/59.2	59/60/59.4
				AG	55/59/56.4	55/58/56.4	55/58/ 56	55/59/56.1
			M10	OMPS	59/60/59.3	59/60/59.4	59/60/59.3	59/60/59.2
				AG	55/59/55.9	55/59/56.1	55/58/ 55.7	55/58/56.2

Fonte: Lian et al., 2006.

Tabela 5 – Comparação entre os resultados do Otimização por Multidão de Partículas Similar (OMPS) e dos Algoritmos Genéticos (AG) com a aplicação dos operadores de cruzamento (C1 - C4) e com os operadores de mutação (M1 - M10) para o problema teste FT10. TP = Tamanho da população para o Otimização por Multidão de Partículas Similar e NG = Número de genes para os Algoritmos Genéticos.

Problema	Solução ótima	TP / NG	Mínimo / Máximo / Média em 10 rodadas					
					C1	C2	C3	C4
FT10	930	1000 / 1500	M1	OMPS	1163/1253/1212.4	1147/1223/1188	1177/1264/1226.9	1202/1261/1232.1
				AG	991/1104/1040.6	960 /1059/1013.9	967/1072/1023.1	969/1061/1021.6
			M2	OMPS	1140/1244/1196.8	1185/1252/1219.4	1182/1256/1223.6	1169/1280/1227.6
				AG	994/1115/1033.1	949 /1083/1014.7	951/1064/1002.5	984/1039/1013.6
			M3	OMPS	1164/1246/1217	1139/1252/1206.1	1164/1248/1217	1193/1267/1235.3
				AG	995/1079/1028.8	969 /1094/1030.4	975/1060/1007.2	970/1064/1018.8
			M4	OMPS	1129/1245/1202.1	1205/1263/1234.4	1155/1240/1213.9	1135/1273/1225.7
				AG	978/1092/1040.8	961 /1039/1005.1	968/1057/1021.2	973/1089/1026.3
			M5	OMPS	1155/1247/1213.8	1173/1256/1214	1136/1266/1226.6	1137/1240/1211.7
				AG	958 /1106/1049	967/1068/1018.3	976/1078/1039.4	961/1081/1022.6
			M6	OMPS	1135/1259/1194.8	1161/1246/1208.8	1122/1253/1213.1	1185/1264/1226.6
				AG	978/1091/1051.5	979/1065/1021.7	971 /1059/1019.6	980/1074/1028.5
			M7	OMPS	1165/1256/1222.7	1189/1253/1215.3	1166/1253/1217.3	1170/1255/1223.5
				AG	977/1066/1025.5	963/1045/1016.2	957 /1076/1011.2	981/1058/1027.9
			M8	OMPS	1191/1225/1212.7	1184/1256/1216.1	1155/1248/1211.1	1177/1252/1235.9
				AG	952 /1070/1022.9	974/1023/1000.1	966/1046/1016.3	966/1046/1016.3
			M9	OMPS	1149/1258/1191.8	1174/1266/1218.5	1147/1244/1210.3	1202/1252/1225.7
				AG	992/1114/1036.6	937 /1103/1022.2	958/1045/10228	995/1060/1019.2
			M10	OMPS	1176/1237/1209.1	1168/1237/1199.3	1132/1230/1196.3	1160/1231/1198.3
				AG	984/1066/1020.7	997/1061/1026.5	967 /1059/1001.8	1007/1069/1032.6

Fonte: Lian et al., 2006.

Tabela 6 – Comparação entre os resultados do Otimização por Multidão de Partículas Similar (OMPS) e dos Algoritmos Genéticos (AG) com a aplicação dos operadores de cruzamento (C1 - C4) e com os operadores de mutação (M1 - M10) para o problema teste FT20. TP = Tamanho da população para o Otimização por Multidão de Partículas Similar e NG = Número de genes para os Algoritmos Genéticos.

Problema	Solução ótima	TP / NG	Mínimo / Máximo / Média em 10 rodadas					
					C1	C2	C3	C4
FT20	1165	1000 / 1500	M1	OMPS	1378/1622/1515.4	1386/1560/1479.8	1378/1589/1481.6	1438/1589/1526.9
				AG	1209/1296/1252.5	1205/1306/1239.6	1191 /1275/1248.8	1200/1321/1241.3
			M2	OMPS	1411/1565/1508.2	1469/1610/1531.9	1411/1612/1480.3	1405/1586/1507.4
				AG	1198/1299/1243.1	1190/1280/1245.6	1199/1328/1240.5	1185 /1284/1224.2
			M3	OMPS	1443/1543/1494.2	1333/1613/1493.9	1407/1609/1525.3	1459/1620/1528.5
				AG	1199/1298/1246.5	1178 /1302/1236.6	1198/1273/1241.5	1190/1289/1230.9
			M4	OMPS	1413/1590/1506	1469/1585/1513.2	1392/1550/1476.4	1450/1670/1520.7
				AG	1210/1284/1232.8	1200/1254/1227.9	1184 /1242/1218.8	1188/1307/1230.7
			M5	OMPS	1392/1557/1484.1	1474/1604/1536.6	1428/1658/1549.5	1447/1601/1517.3
				AG	1190 /1356/1261.5	1192/1270/1233.7	1190/1275/1236.2	1207/1280/1245.9
			M6	OMPS	1419/1579/1482.2	1394/1659/1513.4	1332/1582/1512.4	1463/1608/1532.8
				AG	1184 /1352/1253.4	1199/1298/1245.2	1217/1279/1239.8	1191/1246/1218.2
			M7	OMPS	1484/1627/1552.3	1384/1638/1492.3	1421/1567/1511	1410/1636/1523.8
				AG	1220/1375/1285.1	1192/1276/1237.2	1190/1280/1230.2	1189 /1280/1230.3
			M8	OMPS	1407/1557/1491.3	1370/1587/1478.2	1449/1615/1525.2	1460/1555/1508.1
				AG	1217/1316/1272.2	1178 /1293/1236.1	1182/1299/1227.1	1189/1283/1224.2
			M9	OMPS	1455/1600/1502.3	1421/1576/1524.6	1361/1594/1512.2	1451/1597/1505.7
				AG	1223/1319/1266.3	1182 /1244/1220.3	1195/1314/1244.9	1197/1278/1242.6
			M10	OMPS	1420/1488/1456	1365/1497/1441	1405/1464/1443.5	1392/1475/1.4365
				AG	1193/1364/1279.3	1195/1284/1237.4	1199/1254/1231.6	1182 /1315/1231

Fonte: Lian et al., 2006.

3.2.1 Código de Processamento de Serviços

No Código de Processamento de Serviços (CPS), um escalonamento é representado por uma sequência contendo os índices dos serviços onde o índice referente a cada serviço é repetido na sequência pelo número de vezes correspondente ao número de operações do respectivo serviço. A operação correspondente a cada índice pode ser identificada pelo respectivo serviço e pela posição em que o mesmo aparece, ou seja, o índice que aparece primeiro na sequência definida, dentre todos os índices referentes ao mesmo serviço, representa a primeira operação deste serviço, o índice que aparece em segundo na sequência definida, dentre todos os índices referentes ao mesmo serviço, representa a segunda operação deste serviço, e assim sucessivamente. Por exemplo, a Fig. 8 apresenta uma possível solução para o problema da confecção de roupas masculinas definido no capítulo 2 desta tese, a qual é representada pelo código de processamento de serviços. Nesta sequência, a primeira operação corresponde à primeira operação na ordem de serviço de S6, a segunda operação corresponde a primeira operação na ordem de serviço de S2, a oitava operação corresponde a segunda operação na ordem de serviço de S6, e assim sucessivamente.

CPS = (S6, S2, S4, S3, S5, S1, S2, S6, S3, S6, S1, S4, S6, S5, S1, S4, S5, S2, S6, S5, S3, S5, S3, S3, S6, S5, S1, S4, S4, S1, S2, S4, S2, S2, S1, S3)

Figura 8 – Solução para o problema da confecção de roupas masculinas representada pelo código de processamento de serviços.

Fonte: O autor, 2010.

Em um código de processamento de serviços, a posição de cada operação determina a ordem de prioridade em que ela deve ser processada. Assim sendo, para transformarmos esta sequência na sequência de serviços por máquina, basta pegar cada operação na ordem em que ela aparece na sequência e leva-la à sequência de sua máquina, conforme algoritmo apresentado na Fig. 9. Por exemplo, se substituirmos cada operação pela respectiva máquina em que ela deve ser processada, utilizando como base os dados apresentado na Tab. 1, teremos a sequência de máquinas apresentada na Fig. 10 (a). Posteriormente, se levarmos cada grupo de operações, na ordem em que estas aparecem no código de processamento de serviços, para sua respectiva máquina obteremos a solução representada pelo código de sequência de serviços por máquina, Fig. 10 (b), a qual corresponde a mesma solução anteriormente apresentada nas Figs. 2 e 3.

A conversão do código de sequência de serviços por máquinas para o código de processamento de serviços pode ser conseguida ordenando-se as operações de cada serviço em ordem crescente segundo seu momento inicial de processamento, conforme o algoritmo apresentado na Fig. 12. Na solução apresentada na Fig. 11 (a) cada operação é substituída pelo seu respectivo momento inicial de processamento. Posteriormente o algoritmo de conversão de representação (Fig. 11) é aplicado resultando na sequência de processamento de serviços ilustrada na Fig. 11 (b). Observe que, apesar das soluções apresentadas nas Figs. 10(a) e 11(b) serem aparentemente diferentes quando representadas pelo código de processamento de serviços, elas representam a mesma solução em código de sequência de serviços por máquina.

Sendo:

OPS_i a operação na posição i da ordem de processamento de serviços;

NO é o número total de operações;

$M[OPS_i]$ a máquina em que a operação OPS_i deve ser processada;

NM é o número total de máquinas;

M_{kl} é a l ésima operação na ordem de processamento da máquina $M[k]$;

e $P[k]$ é um ponteiro indicando a posição na ordem de processamento da máquina $M[k]$.

Aplicar o seguinte algoritmo:

Para $m = 1$ até $m = NM$

$P[m] = 1$;

Fim Para

Para $i = 1$ até $i = NO$

$m = M[OPS_i]$

$M_{mP[m]} = OPS_i$

$P[m] = P[m] + 1$

Fim Para

Figura 9 – Algoritmo para transformação do código de processamento de serviços em código de sequência de serviços por máquina.

Fonte: O autor, 2010.

(S6, S2, S4, S3, S5, S1, S2, S6, S3, S6, S1, S4, S6, S5, S1, S4, S5, S2, S6, S5, S3, S5, S3, S3, S6, S5, S1, S4, S4, S1, S2, S4, S2, S2, S1, S3)

(M2, M2, M2, M3, M3, M3, M3, M4, M4, M6, M1, M1, M1, M2, M2, M3, M5, M5, M5, M6, M6, M1, M1, M2, M3, M4, M4, M4, M5, M6, M6, M6, M1, M4, M5, M5)

(a)

M6: S6, S5, S3, S1, S2, S4

M5: S5, S2, S6, S4, S1, S3

M4: S6, S3, S5, S1, S4, S2

M3: S3, S5, S1, S2, S4, S6

M2: S6, S2, S4, S5, S1, S3

M1: S1, S4, S6, S5, S3, S2

(b)

Figura 10 – Solução para o problema da confecção de roupas masculinas representada pelo código de processamento de serviços com sua respectiva sequência de máquinas (a) e pelo código de sequência de serviços por máquina (b).

Fonte: O autor, 2010.

M6: S6, S5, S3, S1, S2, S4

M6: 6, 24, 28, 44, 47, 57

M5: S5, S2, S6, S4, S1, S3

M5: 19, 24, 34, 47, 55, 61

M4: S6, S3, S5, S1, S4, S2

M4: 3, 6, 36, 37, 44, 67

M3: S3, S5, S1, S2, S4, S6

M3: 0, 5, 14, 15, 23, 38

M2: S6, S2, S4, S5, S1, S3

M2: 0, 3, 11, 16, 19, 45

M1: S1, S4, S6, S5, S3, S2

M1: 15, 18, 23, 33, 36, 57

(a)

(S3, S6, S2, S5, S6, S4, S1, S2, S1, S6, S3, S5, S4, S5, S1, S4, S6, S5, S2, S3, S5, S6, S5, S3, S1, S6, S1, S4, S3, S2, S4, S1, S4, S2, S3, S2)

(b)

Figura 11 – Solução para o problema da confecção de roupas masculinas representada pelo código de sequência de serviços por máquina com seus respectivos momentos iniciais de processamento (a) e pelo código de processamento de serviços (b).

Fonte: O autor, 2010.

Sendo:

M_{kl} é a l ésima operação na ordem de processamento da máquina $M[k]$;

OPS_i a operação na posição i da ordem de processamento de serviços;

NO_m é o número total de operações na máquina m ;

$T[j]$ é o tempo inicial de processamento da operação j ;

NM é o número total de máquinas;

e s é o número de operações já ordenadas no código de processamento de serviços.

Aplicar o seguinte algoritmo:

$OPS_i = M_{11}$

$s = 1$;

Para $m = 1$ até $m = NM$

Para $j = 1$ até $j = NO_m$

Se $m = 1$ e $j = 1$

$j = j + 1$

Fim Se

Se $T[OPS_s] < T[M_{mj}]$

$s = s + 1$

$OPS_s = M_{mj}$

Se não

Para $k = 1$ até $k = s$

Se $T[OPS_k] \geq T[M_{mj}]$

Para $l = s$ até $l = k + 1$

$OPS_{l+1} = OPS_l$

Fim Para

$OPS_k = M_{mj}$

$k = k + 1$

Fim Se

Fim Para

$s = s + 1$

Fim Se

Fim Para

Fim Para

Figura 12 – Algoritmo para transformação do código de sequência de serviços por máquina em código de processamento de serviços.

Fonte: O autor, 2010.

3.2.2 Operador de cruzamento C1 com um ponto de corte flutuante

As operações de cruzamento são provenientes da heurística Algoritmos Genéticos onde, a cada iteração, um grupo de soluções genitoras da origem a um novo grupo de soluções filhas, através do cruzamento de pares de soluções genitoras, os quais podem ser selecionado aleatoriamente ou por algum método predefinido. No caso do problema de escalonamento da produção em oficina de máquinas, é apresentada na literatura uma grande variedade de operadores de cruzamento que variam de acordo com o método de cruzamento utilizado e com o tipo de representação do problema. Na presente tese, o operador de cruzamento utilizado é baseado em um dos operadores apresentados por Lian, Gu e Jiao (2006), denominado C1 (corte de um segmento). Este operador de cruzamento pode ser descrito da seguinte forma: um par de pontos de corte é aleatoriamente selecionado ao longo da primeira solução genitora. A subsequência de serviços entre o primeiro e o segundo ponto de corte é copiada e disposta na mesma posição na nova solução a ser gerada. As outras posições da nova solução são preenchidas ordenando-se as operações restantes de acordo com a ordem em que cada legítima operação (representadas pelos respectivos serviços) aparece na segunda solução genitora (Fig. 13).

S6, S2, S4, S3, S5, S1, S2, S6, S3, <u>S6, S1, S4, S6, S5, S1, S4, S5, S2, S6, S5, S3, S5, S3</u> , S3, S6, S5, S1, S4, S4, S1, S2, S4, S2, S2, S1, S3
(1ª solução genitora)
<u>S5, S1, S2, S4, S3</u> , S1, <u>S6</u> , S5, S5, S4, <u>S6</u> , S1, <u>S2</u> , S2, S4, S5, <u>S1, S3</u> , S6, <u>S1, S2, S4</u> , S6, <u>S5, S4</u> , S3, <u>S1, S2, S4</u> , S3, <u>S2</u> , S5, <u>S3</u> , S6, <u>S6, S3</u>
(2ª solução genitora)
S5, S1, S2, S4, S3, S6, S6, S2, S1, <u>S6, S1, S4, S6, S5, S1, S4, S5, S2, S6, S5, S3, S5, S3</u> , S3, S1, S2, S4, S5, S4, S1, S2, S4, S2, S3, S6, S3
solução filha

Figura 13 – Cruzamento de duas soluções representadas pelo código de processamento de serviço pelo operador C1.

Fonte: O autor, 2010.

O problema do operador de cruzamento C1 é que, uma vez que os índices dos serviços são reposicionados, existe a possibilidade de que alguns índices de serviço passem a representar operações diferente das representadas anteriormente. Isso faz com que a solução gerada pelo cruzamento leve pouca informação sobre suas respectivas soluções genitoras, contrariando aos princípios da evolução natural dos indivíduos, nos quais é baseada a heurística Algoritmos Genéticos. Na presente tese utiliza-se um operador de cruzamento definido como C1 com um ponto de corte flutuante no qual apenas um ponto de corte é

selecionado. Posteriormente a subsequência de operações do primeiro cromossomo, que vai da primeira operação à última operação antes do ponto de corte, é copiada e disposta na mesma posição na nova solução a ser gerada e, como no caso de C1, as outras posições da nova solução são preenchidas ordenando-se as operações restantes de acordo com a ordem em que cada legítima operação (representadas pelos respectivos serviços) aparece na segunda solução genitora (Fig. 3.10).

S6, S2, S4, S3, S5, S1, S2, S6, S3, S6, S1, S4, S6, S5, S1, S4, S5, S2, S6, S5, S3, S5, S3, S3, S6, S5, S1, S4, S4, S1, S2, S4, S2, S2, S1, S3
(1ª solução genitora)
S5, S1, S2, S4, S3, S1, S6, S5, S5, S4, S6, S1, S2, S2, S4, S5, S1, S3, S6, S1, S2, S4, S6, S5, S4, S3, S1, S2, S4, S3, S2, S5, S3, S6, S6, S3
(2ª solução genitora)
S6, S2, S4, S3, S5, S1, S2, S6, S3, S6, S1, S4, S6, S5, S1, S4, S5, S2, S6, S5, S3, S1, S1, S2, S4, S5, S4, S1, S2, S4, S3, S2, S5, S3, S6, S3
solução filha

Figura 14 – Cruzamento de duas soluções representadas pelo código de processamento de serviço pelo operador C1 com 1 ponto de corte flutuante.

Fonte: O autor, 2010.

A vantagem deste novo operador é que cada índice continua a representar sua respectiva operação após o cruzamento e as novas soluções geradas carregam informações relevantes de suas respectivas soluções genitoras. É importante observar que cada par de soluções pode gerar pelo menos duas soluções filhas para um ponto de corte selecionado, simplesmente alterando qual das soluções será tratada com a primeira solução genitora. No algoritmo aplicado nesta tese as duas soluções são geradas e entre as duas é escolhida aquela com menor valor do *makespan*.

3.2.3 Operador de mutação M7

No processo de evolução natural, de tempos em tempos, ocorrem alterações nos genes de determinados indivíduos que não estão diretamente associadas a informação genética de seus progenitores. Estas alterações, definidas como mutações, contribuem para que estes indivíduos tragam novas informações genéticas para o seu grupo, permitindo assim uma melhor adaptação ao seu ambiente. No caso dos Algoritmos Genéticos, as operações de mutação geralmente fazem com que o algoritmo vasculhe novos espaços de busca,

aumentando a sua capacidade de fugir de mínimos locais. Existe na literatura uma série de operadores de mutação para o problema de escalonamento de produção em oficina de máquinas, que irão variar de acordo com o método de mutação aplicado e com o tipo de representação do problema. Na presente tese é aplicado o operador de mutação definido por Lian Gu e Jiao (2006) como M7, com reposicionamento de apenas um serviço. Neste operador, um ponto de movimento e um ponto de inserção são escolhidos randomicamente ao longo de uma solução e então a operação posicionada no ponto de movimento é retirada deste ponto e reposicionada no ponto de inserção (Fig. 15).

<p>S6, S2, S4, S3, S5, S1, S2, S6, S3, <u>S6</u>, S1, S4, S6, S5, S1, S4, S5, S2, S6, S5, S3, S5, S3, S3, S6, S5, S1, S4, S4, S1, S2, S4, S2, S2, S1, S3</p> <p>solução antes da mutação</p>
<p>S6, S2, S4, S3, S5, S1, S2, S6, S3, S1, S4, S6, S5, S1, S4, S5, S2, S6, S5, S3, S5, S3, <u>S6</u>, S3, S6, S5, S1, S4, S4, S1, S2, S4, S2, S2, S1, S3</p> <p>solução após mutação</p>

Figura 15 – Mutação de uma solução representada pelo código de processamento de serviço pelo operador M7 com reposicionamento de 1 serviço.

Fonte: O autor, 2010.

Apesar do Otimização por Otimização por Multidão de Partículas Similar apresentar melhores resultados do que o Algoritmos Genéticos, estes resultados ainda não alcançaram o objetivo desejado e adicionalmente, como no caso da Busca Tabu, esta heurística é muito sensível ao ajuste de seus parâmetros, como, por exemplo, à porcentagem de partículas que sofrem operações de mutação em cada iteração, onde pequenos valores contribuem para processo cíclicos ou levam a uma lenta convergência e grandes valores impedem a convergência do algoritmo (Tab. 7).

Tabela 7 – Análise de sensibilidade do *makespan* correspondente à posição final de cada partícula e à melhor posição visitada pelo grupo para o parâmetro “porcentagem de soluções que sofrem mutação por iteração” do método Otimização por Multidão de Partículas Similar, para dez soluções iniciais geradas randomicamente para o problema FT10, com os seguintes parâmetros fixos: operador de cruzamento = C1 com um ponto de corte flutuante; operador de mutação = M1 com reposicionamento de 1 serviço; tamanho da população = 10; número máximo de iterações = 100.000.

Solução	Valor inicial	Porcentagem de soluções que sofrem mutação por iteração										
		0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
1	1668	1469	1003	1028	1011	1177	1220	1403	1522	1301	1731	1382
2	1695	1469	1003	1028	1011	1023	1335	1433	1304	1352	1324	1592
3	1977	1469	1516	1028	1140	1023	1158	1231	1461	1026	1458	1423
4	1807	1469	1003	1028	1166	1147	975	1104	992	1562	1396	1581
5	1749	1469	1003	1028	1011	1023	975	1470	1466	1755	1568	1393
6	1629	1469	1292	1113	1011	1391	1187	1394	1132	1515	991	1581
7	1922	1469	1003	1250	1418	1349	975	1133	1114	1026	1659	1572
8	1783	1469	1003	1146	1011	1561	1000	1433	1494	1532	1323	1735
9	1701	1469	1003	1258	1131	1023	1281	1307	1063	1541	1345	1266
10	1766	1469	1003	1028	1011	1433	1371	1134	1332	1534	1418	1449
Melhor solução		1469	1003	1028	1011	1023	975	1084	992	1026	991	1061

Fonte: O autor, 2010.

3.3 Modelos de hibridização baseados nas Heurísticas de Busca Local

Na seção 1.2.1 foi apresentado um esquema geral para as Heurísticas de Busca Local. Nesse esquema é possível verificar que estas heurísticas podem ser definidas basicamente em três passos que são: (a) geração de um grupo de soluções genitoras iniciais; (b) geração de vizinhança; e (c) seleção do novo grupo de soluções genitoras. A cada passo são utilizados, respectivamente, métodos construtivos, de geração de vizinhança e de seleção para geração do novo grupo de soluções sendo que estes métodos irão variar de acordo com cada heurística específica. Por exemplo, no caso das heurísticas Busca Tabu e Otimização por Multidão de Partículas Similar apresentadas respectivamente nas seções 3.1 e 3.2, as soluções iniciais podem ser geradas randomicamente, sendo que no caso do Busca Tabu uma única solução inicial é gerada enquanto que no caso do Otimização por Multidão de Partículas Similar, é gerado um grupo de soluções iniciais. No Busca Tabu é utilizado o método de geração de vizinhança N5, enquanto que no Otimização por Multidão de Partículas Similar são utilizados métodos de cruzamento e de mutação. Finalmente, para seleção da nova solução genitora, o Busca Tabu utiliza o critério do melhor vizinho não tabu enquanto que o Otimização por Multidão de Partículas Similar seleciona a melhor solução gerada por cada par de soluções genitoras.

Diferente dos métodos construtivos que geram as soluções partindo do zero, os métodos de geração de vizinhança constroem novas soluções pela aplicação de pequenas perturbações no grupo de soluções genitoras, por este motivo na presente tese estes métodos serão tratados como operadores de geração de vizinhança. A idéia de hibridização proposta por Souza de Cursi (2009) e aplicada nesta tese consiste basicamente na geração de uma vizinhança híbrida formada pela aplicação de dois ou mais operadores de geração de vizinhança. A Tab. 8 apresenta alguns dos possíveis operadores de vizinhança, classificados de acordo com sua natureza. Estes operadores podem ser os operadores de geração de vizinhança aplicados nas Heurísticas de Busca Local, conforme anteriormente citado, ou qualquer outro método capaz de transformar um grupo de soluções genitoras em um novo grupo de soluções, através da perturbação das primeiras, como no caso das próprias Heurísticas de Busca Local quando utilizando como suas respectivas soluções iniciais, o grupo de soluções genitoras.

Tabela 8 – Operadores de geração de vizinhança.

	Operadores de troca de operações no caminho crítico	Operadores de cruzamento	Operadores de mutação	Heurísticas de busca local.	
				Novas soluções são geradas pela perturbação em uma única solução	Novas soluções são geradas pelo cruzamento de duas soluções
Determinísticos	N1, N2, N3, N4, N5, N6 (ZHANG et al., 2007)			Busca Tabu	
Não determinísticos		C1, C2, C3, C4 (LIAN et al., 2006)	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10 (LIAN et al., 2006)	Recozimento Simulado	Algoritmos Genéticos, Multidão de Partículas

Fonte: O autor, 2010.

Em seu primeiro trabalho com hibridização de métodos (ver POGU; SOUZA DE CURSI, 1993), Souza de Cursi propôs uma perturbação randômica do Método do Gradiente inspirada pela heurística Recozimento Simulado, para minimização global de uma função custo mapeando a bola de um espaço Euclidiano de dimensão finita. Seu objetivo consistia em, através desta perturbação, assegurar a convergência do método para um mínimo global. Desde então Souza de Cursi vem trabalhando com a hibridização de diferentes métodos para solução de diferentes problemas de otimização (ver SOUZA DE CURSI; POGU, 1994, AUTRIQUE; SOUZA DE CURSI, 1997, SOUZA DE CURSI; MOUATASIM; ELLAIA, 2006, ES-SADEK; ELLAIA; SOUZA DE CURSI, 2009, etc.) e, com base em sua experiência, ele afirma que geralmente a hibridização dos métodos possibilita a geração de algoritmos mais robustos, ou seja, menos sensíveis ao ajuste dos parâmetros e capazes de produzir melhores resultados do que cada um dos métodos separados, desde que os operadores utilizados na hibridização sejam de natureza diferentes. No presente trabalho os métodos Busca Tabu (conforme apresentado na seção 3.1) e Otimização por Multidão de Partículas Similar (conforme apresentado na seção 3.2) foram escolhidos como operadores para geração da vizinhança híbrida por apresentarem diferentes aspectos em diversos sentidos. Essa escolha foi feita com base nas seguintes considerações:

- O método Busca Tabu trabalha sobre cada única solução de forma a conduzi-la a um ponto mínimo que pode ser local ou global. O método Otimização por Multidão de Partículas Similar gera uma iteração entre as soluções do grupo considerado, convergindo os resultados para um ponto ótimo (local ou global) comum a todas as soluções.

- O método Busca Tabu pode ser facilmente apresentado como um método determinístico enquanto que o aspecto randômico é parte essencial e fundamental do método Otimização por Multidão de Partículas Similar.
- O método Busca Tabu com geração de vizinhança pelo método N5 é ainda hoje um dos métodos mais eficazes e eficientes para a solução do problema trabalhado na presente tese.
- Apesar das heurísticas Otimização por Multidão de Partículas Similar e Algoritmos Genéticos apresentarem características similares, o Otimização por Multidão de Partículas Similar tem apresentado, dentro da literatura pesquisada, melhores resultados.

Para geração de vizinhança híbrida são propostos nesta tese três diferentes modelos de hibridização, definidos como Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada. Estes modelos serão apresentados nas próximas secções.

3.3.1 Aplicação Híbrida Sucessiva

Na heurística Aplicação Híbrida Sucessiva (Fig. 16), inicialmente N ($N \geq 1$) soluções são geradas randomicamente. Essas soluções são armazenadas como soluções genitoras e a solução com menor *makespan* é armazenada como melhor solução até então encontrada. Adicionalmente, são escolhidos P ($P \geq 2$) operadores de geração de vizinhança para geração da vizinhança híbrida e é definida a ordem em que estes operadores serão aplicados. Posteriormente, a cada iteração, o primeiro operador é aplicado sobre as N soluções genitoras, gerando um novo conjunto de soluções definido como vizinhança gerada pelo operador 1. Deste conjunto serão selecionadas X soluções sobre as quais será aplicado o segundo operador, gerando um novo conjunto de soluções definido como vizinhança gerada pelo operador 2. Deste conjunto serão selecionadas Y soluções sobre as quais será aplicado o terceiro operador, gerando um novo conjunto de soluções definido como vizinhança gerada pelo operador 3, e assim sucessivamente até a formação da vizinhança gerada pelo operador P . Finalmente, deste conjunto serão selecionadas N soluções as quais serão armazenadas como soluções genitoras e utilizadas para a geração da primeira vizinhança na próxima iteração. A cada formação de vizinhança, a melhor solução é substituída pelo melhor vizinho, caso este último apresente um menor *makespan* do que o do primeiro. O processo continua até que um

dado critério de parada seja satisfeito.

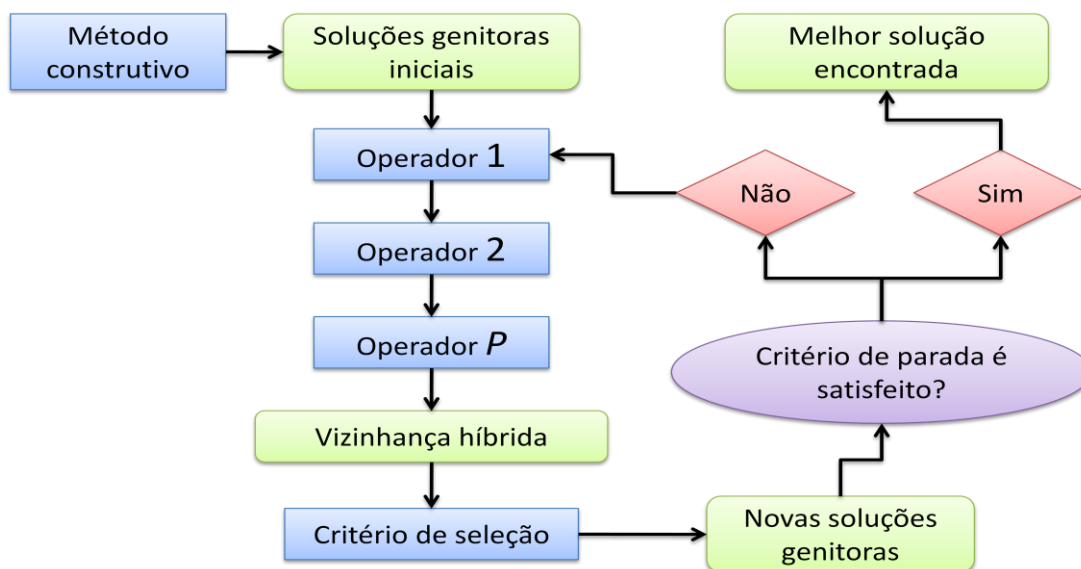


Figura 16 – Esquema para o modelo de hibridização Aplicação Híbrida Sucessiva.

Fonte: O autor, 2010.

3.3.2 Vizinhança Híbrida

Na modelo de hibridização Vizinhança Híbrida (Fig. 17), inicialmente N ($N \geq 1$) soluções são geradas randomicamente. Essas soluções são armazenadas como soluções genitoras e a solução com menor *makespan* é armazenada como melhor solução até então encontrada. Adicionalmente, são escolhidos P ($P \geq 2$) operadores de geração de vizinhança para geração da vizinhança híbrida. Posteriormente, a cada iteração, o primeiro operador é aplicado sobre as N soluções genitoras, gerando um novo conjunto de soluções definido como vizinhança gerada pelo operador 1, o segundo operador é também aplicado sobre as N soluções genitoras, gerando um novo conjunto de soluções definido como vizinhança gerada pelo operador 2, e assim sucessivamente até a formação da vizinhança gerada pelo operador P . Então uma nova vizinhança é formada pela união das vizinhanças geradas por cada um dos operadores e a melhor solução é substituída pelo melhor vizinho, caso este último apresente um menor *makespan* do que o da primeira. Finalmente, desta vizinhança são selecionadas, randomicamente ou através de algum método pré-definido, N soluções as quais serão armazenada como soluções genitoras e utilizadas para a geração das vizinhanças formadas por todos os operadores na próxima iteração. O processo continua até que um dado critério de

parada seja satisfeito.

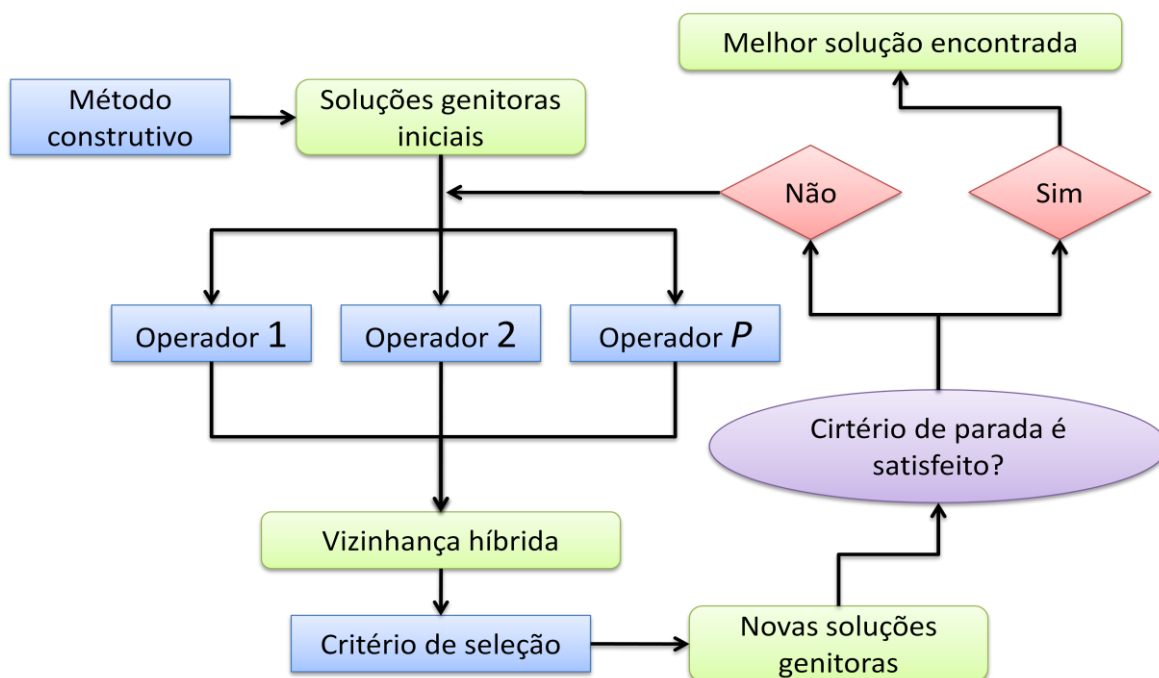


Figura 17 – Esquema para o modelo de hibridização Vizinhaça Híbrida.

Fonte: O autor, 2010.

3.3.3 Vizinhaça Híbrida Melhorada

Na heurística Vizinhaça Híbrida Melhorada (Fig. 18), inicialmente N ($N \geq 1$) soluções são geradas randomicamente. Essas soluções são armazenadas como soluções genitoras e a solução com menor *makespan* é armazenada como melhor solução até então encontrada. Adicionalmente, são escolhidos P ($P \geq 2$) operadores de geração de vizinhaça para geração da vizinhaça híbrida e é definida a ordem em que estes operadores serão aplicados. Posteriormente, a cada iteração, o primeiro operador é aplicado sobre as N soluções genitoras, gerando um novo conjunto de soluções definido como vizinhaça gerada pelo operador 1. Deste conjunto serão selecionadas X soluções sobre as quais será aplicado o segundo operador, gerando um novo conjunto de soluções definido como vizinhaça gerada pelo operador 2. Deste conjunto serão selecionadas Y soluções sobre as quais será aplicado o terceiro operador, gerando um novo conjunto de soluções definido como vizinhaça gerada pelo operador 3, e assim sucessivamente até a formação da vizinhaça gerada pelo operador P . Então uma nova vizinhaça é formada pela união das vizinhaças geradas por cada um dos

operadores e a melhor solução é substituída pelo melhor vizinho, caso este último apresente um menor *makespan* do que o da primeira. Finalmente, desta vizinhança são selecionadas, randomicamente ou através de algum método pré-definido, N soluções as quais serão armazenada como soluções genitoras e utilizadas para a geração da primeira vizinhança na próxima iteração. O processo continua até que um dado critério de parada seja satisfeito.

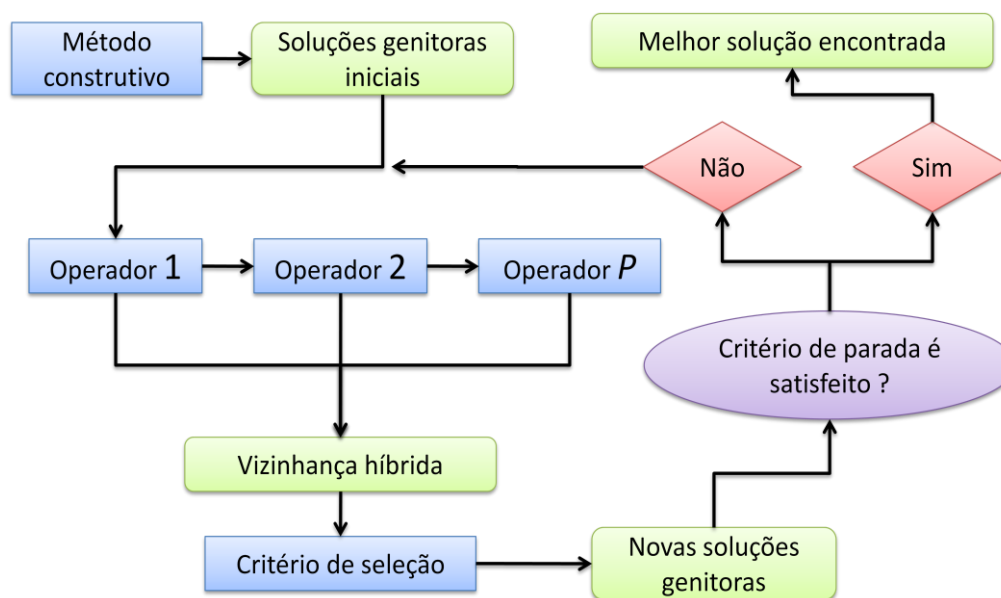


Figura 18 – Esquema para o modelo de hibridização Vizinhança Híbrida Melhorada.

Fonte: O autor, 2010.

Na presente tese, para teste dos modelos de hibridização aqui propostos, são selecionados como operadores de geração de vizinhança o método Busca Tabu (conforme apresentado na seção 3.1) e o método Otimização por Multidão de Partículas Similar (conforme apresentado na seção 3.2), nessa respectiva ordem e na ordem inversa. No caso da Vizinhança Híbrida, a ordem dos operadores teoricamente não deve gerar diferença significativa nos resultados finais contudo, para fins de validação e análise, ambas as ordens serão consideradas. É importante ressaltar que o Otimização por Multidão de Partículas Similar é um método populacional e portanto seu operador deve ser aplicado uma única vez ao conjunto de soluções, já o operador Busca Tabu deve ser aplicado a cada uma das N soluções separadamente (o que pode ser referido como Busca Tabu Paralela - BTP). Adicionalmente, como estes operadores trabalham com diferentes representações de solução - o operador Busca Tabu trabalha com código de sequência de serviços por máquina (CSSM) e o operador Otimização por Multidão de Partículas Similar com código de processamento de serviços (CPS) - é necessário que a representação de cada vizinhança gerada seja

transformada para aplicação do outro operador. Estas transformações devem ser aplicadas seguindo-se os algoritmos apresentados respectivamente nas Figs. 9 e 12.

3.4 Colisão de Partículas Similar

O método Colisão de Partículas foi originalmente desenvolvido por Sacco e Oliveira (2005) com base na física das reações nucleares derivadas da colisão de partículas dentro de um reator nuclear, em particular as reações de absorção e espalhamento. Estas reações aparecem da seguinte forma em um reator: as partículas que atingem núcleos, ou seja, regiões com alta aptidão, são absorvidas e exploram as redondezas enquanto que as partículas que atingem as regiões com baixa aptidão podem ser absorvidas ou espalhadas para outras regiões. Segundo Sacco, Oliveira e Pereira (2006), a sucessão dos eventos de absorção e espalhamento permite simultaneamente uma exploração do espaço de busca completo e uma exploração mais aprofundada dos espaços mais promissores. Conforme Knupp, Silva Neto e Sacco (2009) a grande vantagem do Colisão de Partículas sobre os outros métodos é que apenas um parâmetro precisa ser ajustado em sua aplicação que é o número de iterações. Adicionalmente o Colisão de Partículas apresenta uma estrutura muito simples e de fácil aplicação e tem gerado resultados promissores para solução de problemas de otimização com variáveis contínuas.

O procedimento para aplicação do Colisão de Partículas proposto por Sacco e Oliveira (2005) pode ser apresentado da seguinte forma: primeiramente uma solução inicial é gerada e reservada como melhor solução e solução atual (*Atual_Config*). Posteriormente, a cada iteração a solução atual é perturbada dando origem a uma nova solução (*Nova_Config*) e então os valores correspondente às duas soluções, também referidos como aptidões, são computados de acordo com a função a ser otimizada, e comparados. Caso o valor da nova solução seja melhor do que o valor da solução atual, a nova solução será armazenada como solução atual e sobre essa solução será realizada uma exploração local pela aplicação de pequenas perturbações. No processo de exploração local, caso seja encontrada alguma solução cujo valor seja melhor do que o valor da solução atual, a mesma será armazenada como solução atual. Sendo o valor da nova solução pior do que o valor da solução atual, a nova solução poderá ser explorada ou “espalhada”, ou seja, substituída por uma solução gerada randomicamente, de acordo com uma probabilidade de espalhamento ($P_{\text{espalhamento}}$), a qual

deve ser inversamente proporcional à qualidade de uma dada solução de forma que soluções piores apresentem uma maior probabilidade de espalhamento. No algoritmo original esta probabilidade é definida pelo critério de Metropolis (METROPOLIS et al., 1953). A cada atualização da solução atual, caso a nova solução apresente uma melhor aptidão do que a da melhor solução, a segunda deverá ser substituída primeira. A Fig. 20 apresenta um pseudo código para aplicação do método para problemas de otimização, para problemas de minimização deve-se multiplicar a função objetivo por -1 e inverter a taxa em $P_{\text{espalhamento}}$.

No caso de problemas com variáveis contínuas, as perturbações são geradas através de variações randômicas no valor de cada variável dentro de limites previamente estabelecidos (ver Perturbação (), Fig. 19). As pequenas perturbação são similares às perturbações, diferindo apenas nos limites que são mais estreitos (ver Pequena Perturbação (), na Fig. 21). Quando trabalhamos com problemas de otimização com variáveis discretas, como no caso do problema tratado nesta tese, é preciso definir como essas perturbações serão realizadas. Na presente tese é apresentado o Colisão de Partículas Similar onde as funções Perturbação () e Exploração Local () (Fig. 20) são substituídas, respectivamente, pelos operadores de Perturbação e de Exploração Local.

Perturbação ()

De $i = 0$ a $(\text{dimensão} - 1)$

$Superior = \text{Limite Superior}[i]$

$Inferior = \text{Limite Inferior}[i]$

$Rand = \text{Random}(0,1)$

$Nova_Config[i] := \text{Atual_Config}[i] - ((Superior - \text{Atual_Config}[i]) * Rand) -$
 $((\text{Atual_Config}[i] - Inferior) * (1 - Rand))$

Fim De

Se $(Nova_Config[i] > Superior)$

$Nova_Config[i] := \text{SupLim}[i]$

Se não

Se $(Nova_Config[i] < Inferior)$

$Nova_Config[i] := \text{InfLim}[i]$

Fim Se

Fim Se

Fim

Figura 19 – Função Perturbação.

Fonte: Sacco, Oliveira; Pereira, 2006.

```

Gere uma solução inicial Atual_Config
Melhor Aptidão = Aptidão (Atual_Config)
De n = 0 a # de iterações
    Perturbação ()
    Se Aptidão (Nova_Config) > Aptidão (Atual_Config)
        Se Aptidão (Nova_Config) > Melhor Aptidão
            Melhor Aptidão := Aptidão (Nova_Config)
        Fim Se
        Atual_Config := Nova_Config
        Exploração local ()
    Se não
        Espalhamento ()
    Fim Se
Fim De

Exploração local ()
    De n = 0 a # de iterações
        Pequena Perturbação ()
        Se Aptidão (Nova_Config) > Aptidão (Atual_Config)
            Atual_Config := Nova_Config
        Fim Se
    Fim De
retorna

Espalhamento ()

$$P_{\text{espalhamento}} = 1 - \frac{\text{Aptidão}(\text{Nova\_Config})}{\text{MelhorAptidão}}$$

    Se  $P_{\text{espalhamento}} > \text{random}(0, 1)$ 
        Atual_Config = Solução randômica
    Se não
        Exploração local ()
    Fim Se
Retorna

```

Figura 20 – Pseudo código para o Multidão de Partículas.

Fonte: Sacco, Oliveira; Pereira, 2006.

```

Pequena Perturbação ()
  De i = 0 a (dimensão - 1)
    Superior = Random (1.0, 1.2) * Atual_Config[i]
    Se ( Superior > Limite Superior [i])
      Superior := Limite Superior [i]
    Fim Se
    Inferior = Random (0.8, 1.0) * Atual_Config[i]
    Se ( Inferior > Limite Inferior [i])
      Inferior := Limite Inferior [i]
    Fim Se
    Rand = Random (0,1)
    Nova_Config [i] := Atual_Config [i] - ((Superior - Atual_Config[i])*Rand) -
      ((Atual_Config[i] - Inferior)*(1 - Rand))
  Fim De
Fim

```

Figura 21 – Função Pequena Perturbação.

Fonte: Sacco, Oliveira; Pereira, 2006.

Como operadores de Perturbação, que têm como finalidade levar o algoritmo para novos espaços de busca, são sugeridos os mesmo operadores de mutação utilizados no Algoritmos Genéticos, como por exemplo os operadores M1 a M10 apresentados por Lian, Gu e Jiao (2006). Como alternativa, caso o algoritmo Colisão de Partículas Similar seja aplicado simultaneamente à todos os indivíduos de uma população inicial, conforme proposto a seguir, pode-se utilizar os mesmos operadores de cruzamento do Algoritmos Genéticos, como por exemplo os operadores C1 a C4 apresentados por Lian, Gu e Jiao (2006), sobre os pares de soluções. No caso dos operadores de Exploração Local, que têm como finalidade explorar as redondezas de uma dada solução, sugere-se os diversos algoritmos construídos com base nas heurísticas de busca local. Na presente tese a heurística Colisão de Partículas Similar aqui proposta (Fig. 22) é aplicada sobre um conjunto de soluções iniciais geradas randomicamente e nesse caso ela é definido como Multi Colisão de Partículas Similar. Como operador de perturbação é aplicado como teste o operador de mutação M7, com reposicionamento de apenas um serviço, conforme adotado nesta tese para aplicação do Otimização por Multidão de Partículas Similar e, como operador de exploração local, é aplicado como teste o método Busca Tabu, conforme apresentado na seção 3.2. A Fig. 22

apresenta esquema geral para o Colisão de Partículas Similar, proposto nesta tese.

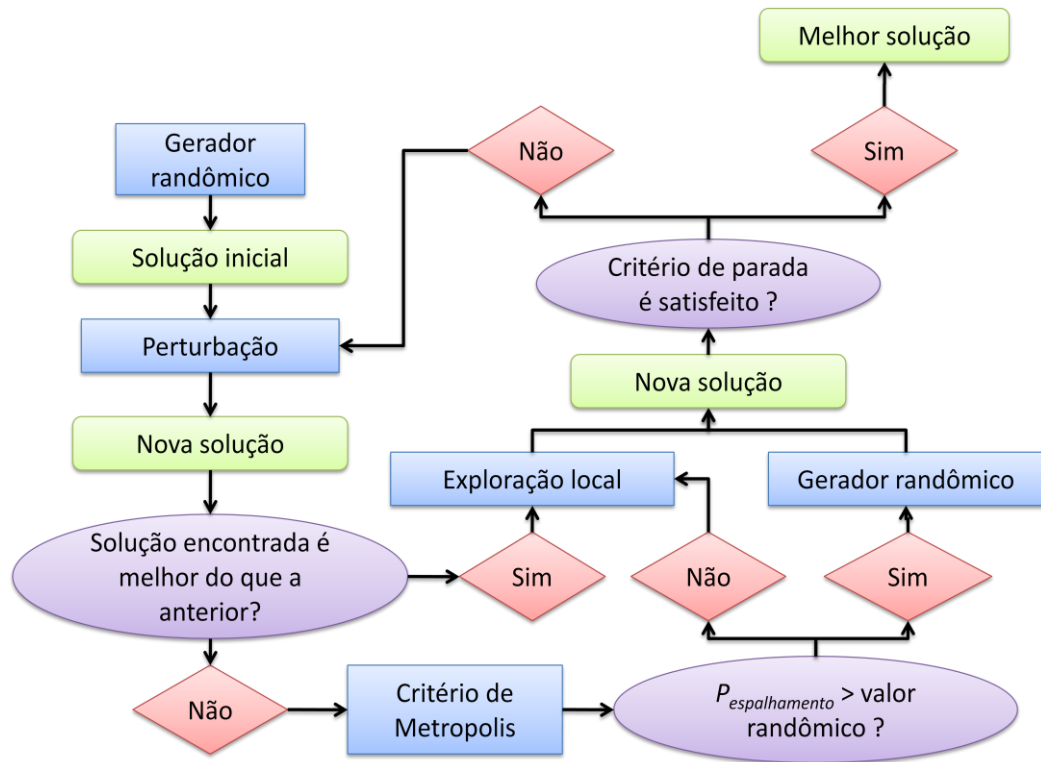


Figura 22 – Colisão de Partículas Similar.

Fonte: O autor, 2010.

4 RESULTADOS E DISCUSSÕES

Para teste dos algoritmos propostos nesta tese foram utilizados os problemas FT6 (com 6 máquinas e 6 serviços), FT10 (com 10 máquinas e 10 serviços) e FT20 (com 5 máquinas e 20 serviços) propostos por Fisher e Thompson (1963) e os problemas ABZ5 e ABZ6 (com 10 máquinas e 10 serviços), e ABZ7, ABZ8 e ABZ9 (com 15 máquinas e 20 serviços) propostos por Adams, Balas e Zawack (1988) (para maior detalhamento dos problemas ver anexo 1). Para cada problema foram geradas randomicamente 12 soluções iniciais e as mesmas foram utilizadas como soluções iniciais para todos os testes aplicados. No caso dos modelos de hibridização propostos, para amenizar a distorção gerada pelo aspecto randômico, cada teste aplicado foi repetido 5 vezes.

Inicialmente foi gerada uma análise de sensibilidade para o problema FT10 sobre os modelos de hibridização Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada com os operadores Busca Tabu Paralela e Otimização por Multidão de Partículas Similar, nesta ordem e com a ordem invertida, variando-se os valores dos parâmetros “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu e “número de iterações” do método Otimização por Multidão de Partículas Similar entre 10, 100, 1.000 e 10.000 e do parâmetro “porcentagem de partículas que sofrem mutação por iteração” do método Multidão de Partículas Similar entre 20% e 80% (Tab. 9). Desta análise verificou-se que todos os modelos são capazes de convergir para soluções ótimas, ou pelo menos melhores do que as geradas pela simples aplicação dos métodos Busca Tabu e Multidão de Partículas Similar, sendo que os modelos apresentaram uma forte sensibilidade ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” do método Otimização por Multidão de Partículas Similar. Foi observado que o método Otimização por Multidão de Partículas Similar gera rapidamente uma conversão das soluções para um único resultado final sendo que esta conversão está diretamente vinculada ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração”. Quanto menor é o valor estabelecido para esta variável maior será o número de soluções convergidas à solução final. Quando o operador Busca Tabu Paralela é aplicado sobre as soluções geradas pelo operador Otimização por Multidão de Partículas Similar, caso exista uma convergência de um grande número de soluções, o método Busca Tabu acaba sendo aplicado várias vezes sobre uma mesma solução. Esse comportamento explica a obtenção de melhores resultados quando

se ajusta o parâmetro “porcentagem de partículas que sofrem mutação por iteração” com valores mais altos.

Tabela 9 – Resumo dos resultados obtidos pela aplicação dos modelos de hibridização à 10 soluções iniciais geradas randomicamente para o problema FT10 (com os seguintes parâmetros fixos: número de ciclos = 5 – Busca Tabu: método de geração de vizinhança = N5; e tamanho da lista tabu = 8; – Multidão de Partículas Similar: método de cruzamento = C1 com 1 ponto de corte flutuante; método de mutação = M7; percentagem de partículas que sofrem mutação = 20% ou 80%).

Nº de iterações	Método	Aplicação Híbrida Sucessiva				Vizinhança Híbrida		Vizinhança Híbrida Melhorada			
		MPS-BT 20%	MPS-BT 80%	BT-MPS 20%	BT-MPS 80%	MPS-BT 20%	MPS-BT 80%	MPS-BT 20%	MPS-BT 80%	BT-MPS 20%	BT-MPS 80%
BT : 10 MPS : 10	Makespan	1043	962	1027	977	1027	1011	1041	977	1010	982
	NSolComp.	8722	14519	10067	16560	10941	13218	9493	60045	9086	13336
	CPU (s)	3	4	3	6	3	3	3	16	2	4
BT : 100 MPS : 100	Makespan	983	942	955	958	965	963	977	942	964	958
	NSolComp.	68056	97256	72524	95796	67162	82134	60045	91369	64849	81716
	CPU (s)	19	26	21	29	18	22	16	24	17	21
BT : 1.000 MPS : 1.000	Makespan	945	938	945	935	945	943	945	934	945	945
	NSolComp.	627445	794550	628900	798314	665734	796400	630332	780278	630800	769781
	CPU (s)	162	207	184	207	172	206	163	202	162	199
BT : 10.000 MPS : 10.000	Makespan	940	930	937	930	937	930	940	930	930	930
	NSolComp.	5891463	1381868	5867281	2372932	5865063	3021198	5847563	1381868	4908378	2264556
	CPU (s)	1513	355	1506	614	1506	776	1506	357	1260	584
BT : 10.000 MPS : 1	Makespan	930	934	934	934	930	937	934	937	934	934
	NSolComp.	1731952	4542479	3851663	4463389	3330616	4081269	3705411	3893905	3567571	3576298
	CPU (s)	452	1182	1002	1162	872	1069	965	1014	931	934

Fonte: O autor, 2010.

Posteriormente os modelos de hibridização foram aplicados com apenas uma iteração do método Otimização por Multidão de Partículas Similar e foi observado que quando estes modelos são aplicados desta forma, combinam a propriedade de convergência global do Otimização por Multidão de Partículas Similar com a propriedade de convergência local do método Busca Tabu minimizando a sensibilidade do algoritmo aos diferentes parâmetros e gerando bons resultados.

Nas próximas secções são apresentados e discutidos os resultados de análises da variação do *makespan* com relação aos diferentes parâmetros. A seção 4.1 apresenta uma análise de sensibilidade do *makespan* aos valores dos parâmetros “número máximo de iterações sem melhora do *makespan*” e “tamanho da lista tabu” do método Busca Tabu e introduz o conceito de ponto de saturação do primeiro parâmetro citado. A seção 4.2 apresenta uma análise de sensibilidade do *makespan* à variação dos parâmetros “número de ciclos” para os modelos de hibridização Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança

Híbrida Melhorada, aplicados com apenas uma iteração do método Otimização por Multidão de Partículas Similar, e para o Multi Colisão de Partículas Similar, comparando e verificando a convergência destes modelos. As secções 4.3 e 4.4 apresentam, respectivamente, análises de sensibilidade do *makespan* à variação dos parâmetros “tamanho da lista tabu” e “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu para os modelos de hibridização Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada, aplicados com apenas uma iteração do método Otimização por Multidão de Partículas Similar. Adicionalmente é feita esta mesma análise para o Multi Colisão de Partículas Similar e os diferentes modelos de hibridização são comparados. Finalmente, a secção 4.5 apresenta uma análise de sensibilidade do *makespan* à variação do parâmetro “porcentagem de partículas que sofrem mutação por iteração” do método Otimização por Multidão de Partículas Similar para os modelos de hibridização Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada, aplicados com apenas uma iteração do método Multidão de Partículas Similar. As tabelas com os valores dos resultados dos teste aplicados são apresentadas no anexo 2.

4.1 Análise de sensibilidade do *makespan* ao valor dos parâmetros “número máximo de iterações sem melhora do *makespan*” e “tamanho da lista tabu” para o método Busca Tabu.

As Tabs. 28, 30, 51, 53, 65, 71, 77 e 83 do anexo 2, apresentam para as doze soluções iniciais geradas para os problemas testes FT6, FT10, FT20, ABZ5, ABZ6, ABZ7, ABZ8 e ABZ9, respectivamente, análises da variação do *makespan* quando o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” é ajustado entre 100, 1.000, 10.000, 100.000 e 1.000.000. Através destas tabelas é possível verificar que, para cada conjunto de soluções, o *makespan* decresce com o aumento do valor do parâmetro analisado até um determinado ponto de saturação (PS) a partir do qual nenhum aumento no valor deste parâmetro é capaz de gerar uma melhora do *makespan* para qualquer uma das soluções consideradas. Estas tabelas também demonstram que o valor deste ponto de saturação está diretamente associado ao tamanho do problema. Na Fig. 23 é apresentada uma análise da variação do *makespan* ao aumento do valor do parâmetro “número máximo de iterações sem melhora do *makespan*” para o problema FT10 onde verifica-se que para este problema teste

PS = 10.000. O mesmo ponto de saturação é verificado para os problemas ABZ5⁴ e ABZ6, os quais apresentam o mesmo número de máquinas e serviços que o FT10. Para o problema FT20 foi verificado que PS = 1.000 e para os problemas com 15 máquinas e 20 serviços (ABZ7, ABZ8 e ABZ9) foi verificado que PS = 100.000. No caso do problema FT6, como todas as soluções convergiram para um ótimo global com pouquíssimas iterações, ele não será utilizado para análise dos modelos de hibridização propostos nesta tese.

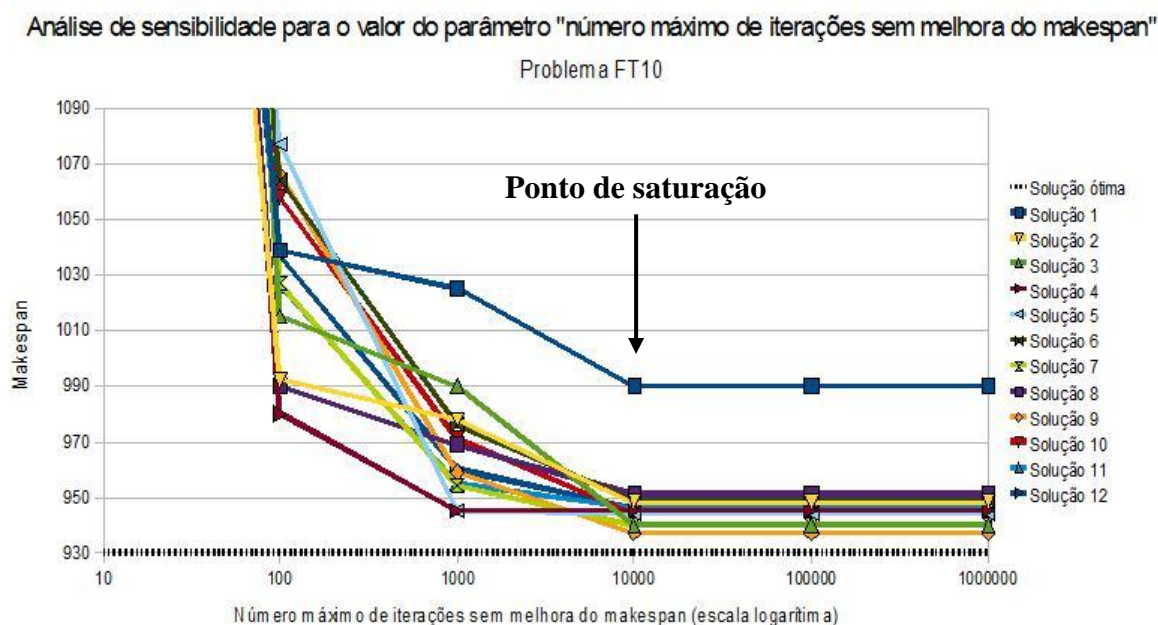


Figura 23 – Análise de sensibilidade do makespan ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu para o problema FT10 (com os seguintes parâmetros fixos: método de geração de vizinhança = N5; e tamanho da lista tabu = 8).

Fonte: O autor, 2010.

Outra importante observação proveniente desta análise é que, apesar do número de soluções computadas ter aumentado com o aumento do valor do parâmetro “número máximo de iterações sem melhora do *makespan*”, a média das soluções computadas por iteração foi bastante reduzida com o aumento desta variável para todos os problemas testes, sendo que esta redução foi ainda mais significativa para os problemas maiores (Fig. 24). Também foi verificado que o número de soluções computadas por unidade de tempo é reduzido significativamente a medida em que aumenta-se o tamanho do problema (número de máquinas e serviços) e portanto o número de soluções possíveis (Fig. 25) de forma que, a medida em que se aumenta o tamanho do problema, aumenta-se também significativamente o tempo de processamento necessário para computar um determinado número de soluções.

⁴No caso do ABZ5, 11 das 12 soluções definem o ponto de saturação.

Como o número de soluções computadas para um determinado valor definido para o parâmetro “número máximo de iterações sem melhora do *makespan*” aumenta significativamente com o aumento do tamanho do problema, como resultado temos um aumento significativo no tempo de processamento quando trabalhamos com problema maiores.

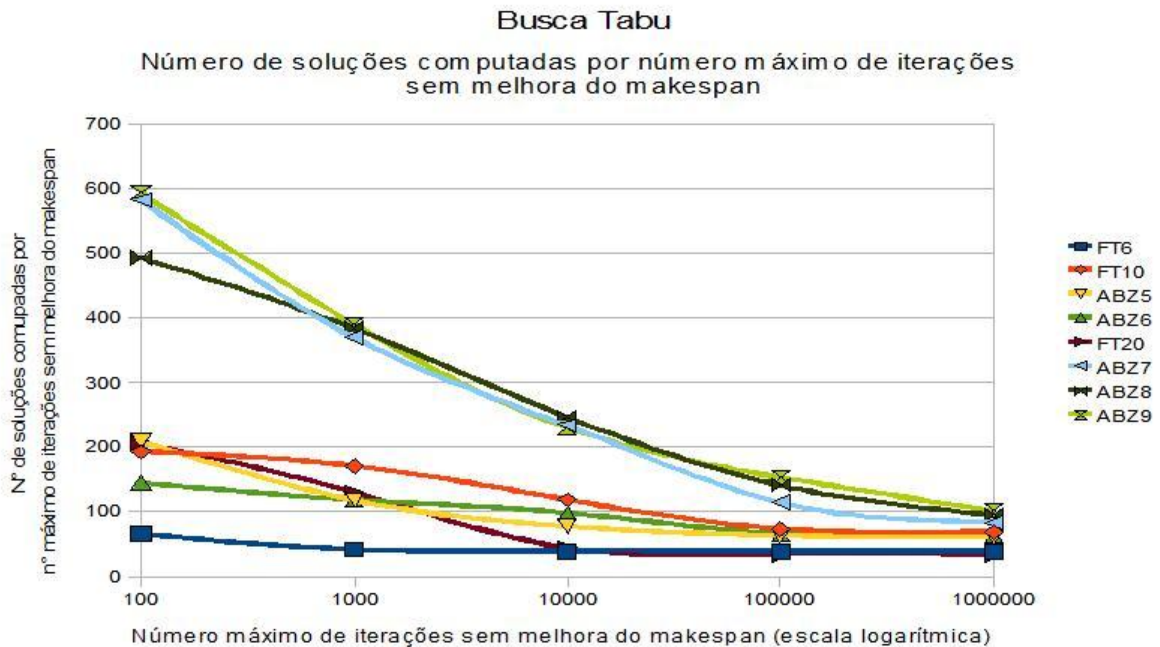


Figura 24 – Número de soluções computadas por número máximo de iterações sem melhora do *makespan*.

Fonte: O autor, 2010.

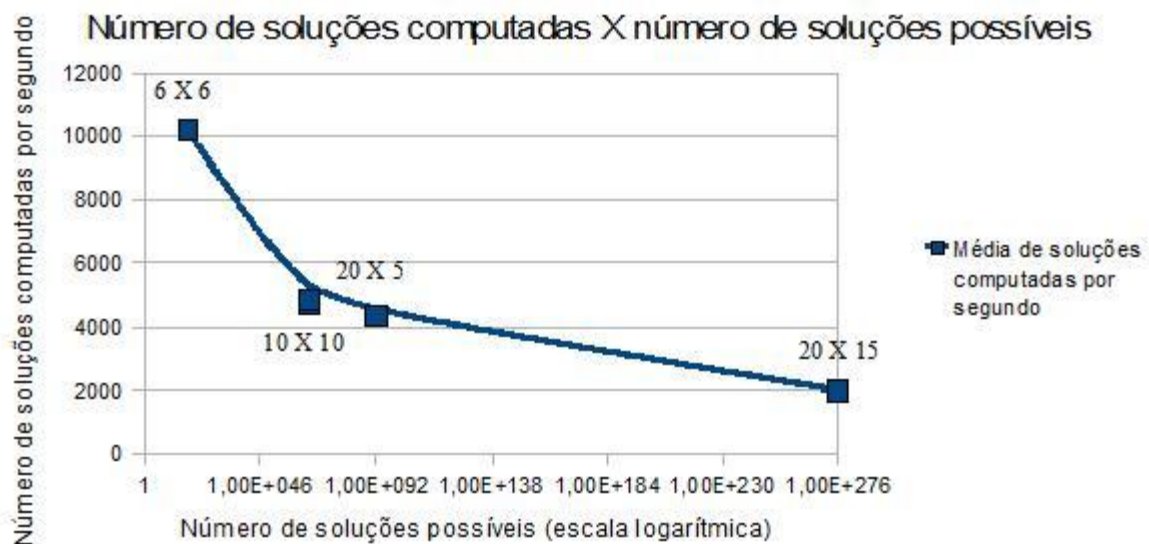


Figura 25 – Análise de sensibilidade do número de soluções computadas à variação do tamanho do problema.

Fonte: O autor, 2010.

Por exemplo: nos testes realizados, quando o Busca Tabu foi aplicado ao problema ABZ5 (com 10 máquinas e 10 serviços), fixando-se o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” em 1.000.000, foram computadas 60.760.918 soluções em aproximadamente 36 minutos, no caso do ABZ7 (com 15 máquinas e 20 serviços), 83.340.148 soluções foram computadas em quase 2 horas, ou seja, um pequeno aumento no tamanho do problema resultou em um aumento de 1:20h no tempo de computação sendo que esta diferença se torna cada vez maior a medida em que se aumenta ainda mais o tamanho do problema. No caso dos modelos de hibridização propostos nesta tese, este aumento será ainda mais significativo uma vez que o Busca Tabu é repetido em cada ciclo.

As Tabs. 29, 31, 52, 54, 66, 72, 78 e 84, do anexo 2, apresentam para as mesmas doze soluções iniciais para os problema testes FT6, FT10, FT20, ABZ5, ABZ6, ABZ7, ABZ8 e ABZ9, respectivamente, uma análise da variação do *makespan* quando o valor do parâmetro “tamanho da lista tabu” é ajustado entre 6 e 15, e o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” é fixado no ponto de saturação de cada problema (1.000 para FT20, 10.000 para os problemas FT10, ABZ5 e ABZ6 e 100.000 para os problemas ABZ7, ABZ8 e ABZ9). Por estas tabelas é possível verificar que o valor deste parâmetro influi diretamente no *makespan* encontrado sendo que não existe nenhuma relação de proporcionalidade entre a variação do *makespan* e a variação do parâmetro analisado (Fig. 26).

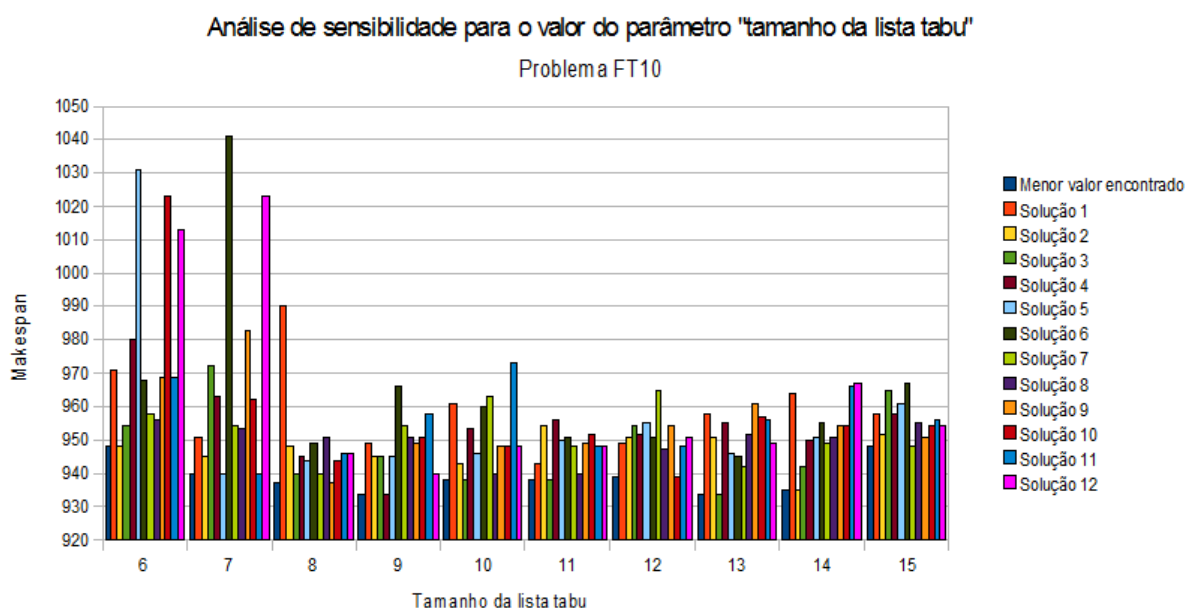


Figura 26 –Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” do método Busca Tabu para o problema FT10 (com os seguintes parâmetros fixos: método de geração de vizinhança = N5; e número máximo de iterações sem melhora do *makespan* = 10.000).

Fonte: O autor, 2010.

4.2 Análise de convergência

Para análise de convergência dos modelos de hibridização apresentados nesta tese, inicialmente foi gerada, para o problema FT20, uma análise de sensibilidade do *makespan* à variação do parâmetro “número de ciclos”, fixando-se o valor do parâmetro “número máximo de ciclos” dos modelos de hibridização em 100 e variando-se o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do operador Busca Tabu Paralela entre 100 e 1.000. Posteriormente foi feita para o mesmo problema uma comparação entre os resultados encontrados pelos modelos de hibridização e pelo método Busca Tabu. Finalmente, foi realizada uma análise de convergência para os modelos de hibridização Aplicação Híbrida Sucessiva e Multi Colisão de Partículas Similar, para os problemas FT10, ABZ5, ABZ6, ABZ7, ABZ8 e ABZ9, fixando-se o valor do parâmetro “número máximo de ciclos” dos modelos de hibridização em 1.000 e o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do operador Busca Tabu Paralela em 10.000 para os problemas FT10, ABZ5 e ABZ6 e em 100.000 para os problemas ABZ7, ABZ8 e ABZ9. Os resultados das análises geradas com o problema FT20 para os modelos de hibridização Aplicação Híbrida Sucessiva, Vizinhança Híbrida, Vizinhança Híbrida Melhorada e Multi Colisão de Partículas Similar são apresentados, respectivamente, nas seções 4.2.1, 4.2.2, 4.2.3 e 4.2.4. A comparação entre estes resultados e os resultados gerados pelo Busca Tabu é apresentada na seção 4.2.5. As análises de convergência para os problemas FT10 e ABZ5 a ABZ9 são apresentadas na seção 4.2.6.

4.2.1 Resultados da análise de convergência para o modelo de hibridização Aplicação Híbrida Sucessiva

Para o caso em que o modelo Aplicação Híbrida Sucessiva foi aplicado ao problema FT20 com os operadores Busca Tabu Paralela e Otimização por Multidão de Partículas Similar, nesta ordem (BTP/MPS), fixando-se o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do operador Busca Tabu em 100, foi verificado que o algoritmo convergiu em todas as 5 rodadas testadas com no máximo 55 ciclos (Fig. 27). Também foi verificado que o aumento do valor desta variável entre 100 e 700 resultou em

uma aceleração da convergência sendo que, quando esta variável foi ajustada entre 700 e 1.000, o algoritmo convergiu para a solução ótima em todas as 5 rodadas com apenas 1 ciclo (ver Tab. 10).

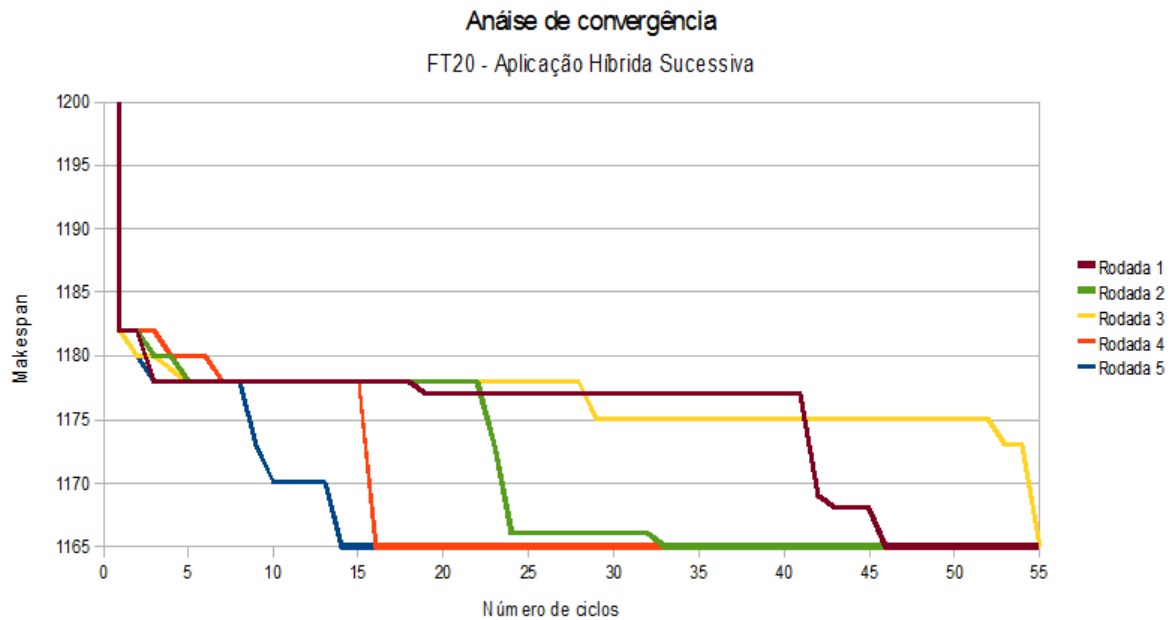


Figura 27 – Análise de convergência do *makespan* para o problema FT20 e o modelo de hibridização Aplicação Híbrida Sucessiva com a ordem dos operadores de geração de vizinhança Busca Tabu – Otimização por Multidão Similar de Partículas.

Fonte: O autor, 2010.

Tabela 10 – Mínimo, máximo e média, em 5 rodadas, do número de ciclos em que o modelo de hibridização Aplicação Híbrida Sucessiva convergiu para a solução ótima do problema FT20.

		Número máximo de iterações sem melhora do <i>makespan</i> do operador Busca Tabu									
		100	200	300	400	500	600	700	800	900	1000
BTP / MPS	min	14	8	5	2	3	3	1	1	1	1
	max	55	24	10	5	6	4	1	1	1	1
	med	32,8	14	7,8	3,6	4,8	3,4	1	1	1	1
MPS / BTP	min	7	3	1	1	3	4	1	1	1	1
	max	22	19	14	6	6	5	4	3	3	3
	med	13,4	9,4	5,8	4	4,4	4,4	2,8	2,2	2,2	2

Fonte: O autor, 2010.

Quando a ordem dos operadores foi invertida (MPS/BTP), foi observada uma convergência significativamente mais rápida quando o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do operador Busca Tabu foi fixado em 100 (Fig. 28)

e, como no caso anterior, o aumento do valor desta variável gerou um aumento na velocidade de convergência, contudo, a diferença da velocidade de convergência entre as duas ordenações consideradas foi diminuída a medida em que o valor desta variável foi aumentado entre 100 e 500 sendo que, quando o este valor foi ajustado entre 600 e 1.000, o modelo aplicado com a ordem dos operadores BTP/MPS apresentou uma convergência mais rápida (ver Tab. 10).

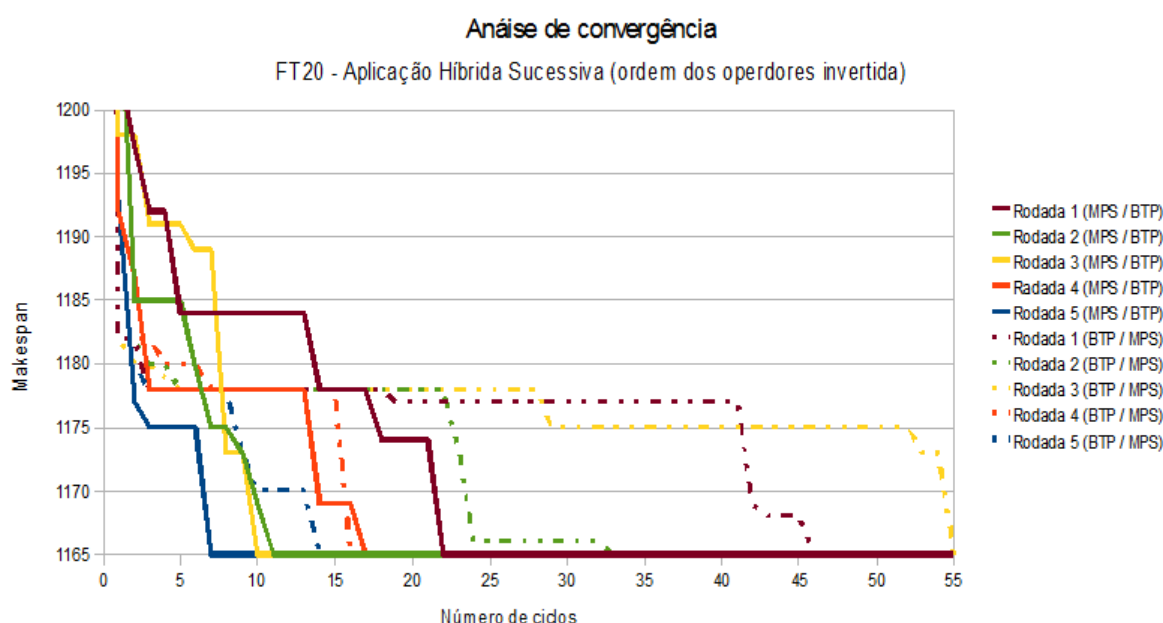


Figura 28 – Análise de convergência do *makespan* para o problema FT20 e o modelo de hibridização Aplicação Híbrida Sucessiva com a ordem dos operadores de geração de vizinhança Otimização por Multidão Similar de Partículas - Busca Tabu.

Fonte: O autor, 2010.

4.2.2 Resultados da análise de convergência para o modelo de hibridização Vizinhança Híbrida

No caso em que o modelo de hibridização Vizinhança Híbrida foi aplicado ao problema FT20, com os operadores Busca Tabu Paralela e Multidão de Partículas Similar, nesta ordem (BTP/MPS), fixando-se o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do operador Busca Tabu Paralela em 100 e 200, foi verificado que houve convergência do *makespan* para a solução ótima em 4 das 5 rodadas (Fig. 29). Quando o valor desta variável foi fixado em 300 houve uma convergência para solução ótima em todas

as 5 rodadas testadas com no máximo 43 ciclos, e o aumento do valor desta variável entre 300 e 600 resultou em uma variação aleatória da velocidade de convergência sendo que, quando esta variável foi ajustada entre 700 e 1.000, como no caso do modelo de hibridização Aplicação Híbrida Sucessiva, o algoritmo convergiu para a solução ótima em todas as 5 rodadas com apenas 1 ciclo (ver Tab. 11). Quando a ordem dos operadores foi invertida (MPS/BTP) e o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do operador Busca Tabu Paralela foi fixado em 100, 4 das 5 rodadas não convergiram para uma solução ótima dentro do limite máximo definido para o número de ciclos (Fig. 30) contudo, o aumento do valor desta variável entre 200 e 600 gerou um aumento na velocidade de convergência e, como no caso anterior, quando esta variável foi ajustada entre 700 e 1.000 o algoritmo convergiu para a solução ótima em todas as 5 rodadas com apenas 1 ciclo (ver Tab. 11). Em geral verificou-se que os resultados gerados pelo modelo de hibridização Aplicação Híbrida Sucessiva foram significativamente melhores dos que os resultados obtidos pelo modelo de hibridização Vizinhança Híbrida.

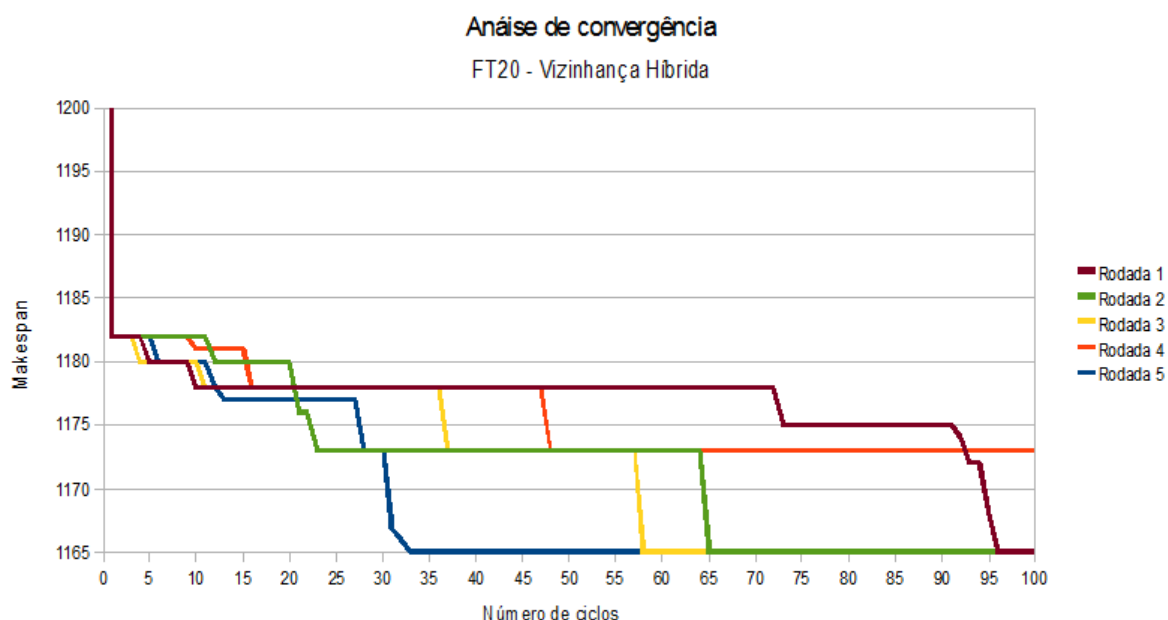


Figura 29 – Análise de convergência do *makespan* para o problema FT20 e o modelo de hibridização Vizinhança Híbrida com a ordem dos operadores de geração de vizinhança Busca Tabu – Otimização por Multidão Similar de Partículas.

Fonte: O autor, 2010.

Tabela 11 – Mínimo, máximo e média, em 5 rodadas, do número de ciclos em que o modelo Vizinhança Híbrida convergiu para a solução ótima do problema FT20. (NC = solução não convergiu antes de 100 ciclos - para cálculo da média, quando a solução não converge para o ótimo, são considerados 100 ciclos).

		Número máximo de iterações sem melhora do <i>makespan</i> do operador Busca Tabu									
		100	200	300	400	500	600	700	800	900	1000
BTP / MPS	min	33	5	2	3	10	3	1	1	1	1
	max	NC	NC	43	58	96	41	1	1	1	1
	med	70,4	33	14,6	16,6	31	16,4	1	1	1	1
MPS / BTP	min	81	5	8	4	3	3	1	1	1	1
	max	NC	65	30	39	10	16	1	1	1	1
	med	96,2	20,8	14,8	15,2	7,2	7,4	1	1	1	1

Fonte: O autor, 2010.

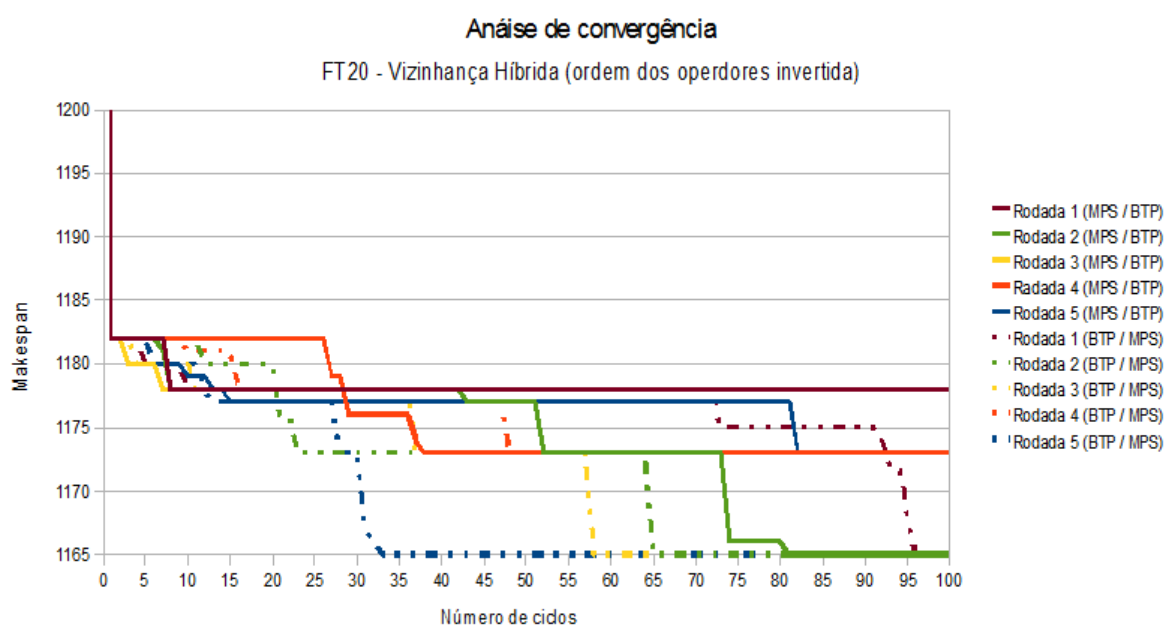


Figura 30 – Análise de convergência do *makespan* para o problema FT20 e o modelo de hibridização Vizinhança Híbrida com a ordem dos operadores de geração de vizinhança Otimização por Multidão Similar de Partículas - Busca Tabu.

Fonte: O autor, 2010.

4.2.3 Resultados da análise de convergência para o modelo de hibridização Vizinhança Híbrida Melhorada

Inicialmente, quando o modelo de hibridização Vizinhança Híbrida Melhorada foi aplicado ao problema FT20, com os operadores Busca Tabu Paralela e Otimização por

Multidão de Partículas Similar, nesta ordem (BTP/MPS), fixando-se o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do operador Busca Tabu Paralela em 100, foi verificado que em apenas 1 das 5 rodadas o algoritmo não convergiu para a solução ótima dentro do limite máximo de ciclos estabelecido sendo que, nas outras 4 rodadas o algoritmo convergiu em menos de 25 ciclos (Fig. 31). O aumento do valor desta variável entre 200 e 700 resultou em uma melhora na velocidade de convergência do algoritmo sendo que, quando esta variável foi ajustada entre 700 e 1.000, como no caso dos modelos de hibridização Aplicação Híbrida Sucessiva e Vizinhança Híbrida, o algoritmo convergiu para a solução ótima em todas as 5 rodadas com apenas 1 ciclo (ver Tab. 12). Quando a ordem dos operadores foi invertida (MPS/BTP) e o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do operador Busca Tabu foi fixado em 100, observou-se que houve uma grande variação na velocidade de convergência para as 5 rodadas (Fig. 32). O aumento do valor desta variável entre 200 e 1.000 gerou um aumento na velocidade de convergência sendo que o mesmo sofreu visivelmente uma forte influência do fator randômico. Em geral verificou-se que os resultados gerados pelo modelo de hibridização Aplicação Híbrida Sucessiva foram também significativamente melhores dos que os resultados obtidos pelo modelo de hibridização Vizinhança Híbrida Melhorada. Diferente do que ocorreu como os outros dois modelos de hibridização anteriores, no caso deste último modelo a inversão da ordem dos operadores de geração vizinhança não resultou em melhoras dos resultados encontrados.

Tabela 12 – a Mínimo, máximo e média, em 5 rodadas, do número de ciclos em que o modelo de hibridização Vizinhança Híbrida Melhorada convergiu para a solução ótima do problema FT20. (NC = solução não convergiu antes de 100 ciclos - para cálculo da média, quando a solução não converge para o ótimo, são considerados 100 ciclos).

		Número máximo de iterações sem melhora do <i>makespan</i> do operador Busca Tabu									
		100	200	300	400	500	600	700	800	900	1000
BTP / MPS	min	14	9	8	4	5	6	1	1	1	1
	max	NC	61	NC	26	27	21	1	1	1	1
	med	35,6	31,2	34,4	9	12,6	12,6	1	1	1	1
MPS / BTP	min	2	3	1	2	1	1	1	1	1	1
	max	NC	16	15	39	51	17	3	14	5	5
	med	54	8,8	7,4	12,2	18,6	5,6	2	4,4	1,8	1,8

Fonte: O autor, 2010.

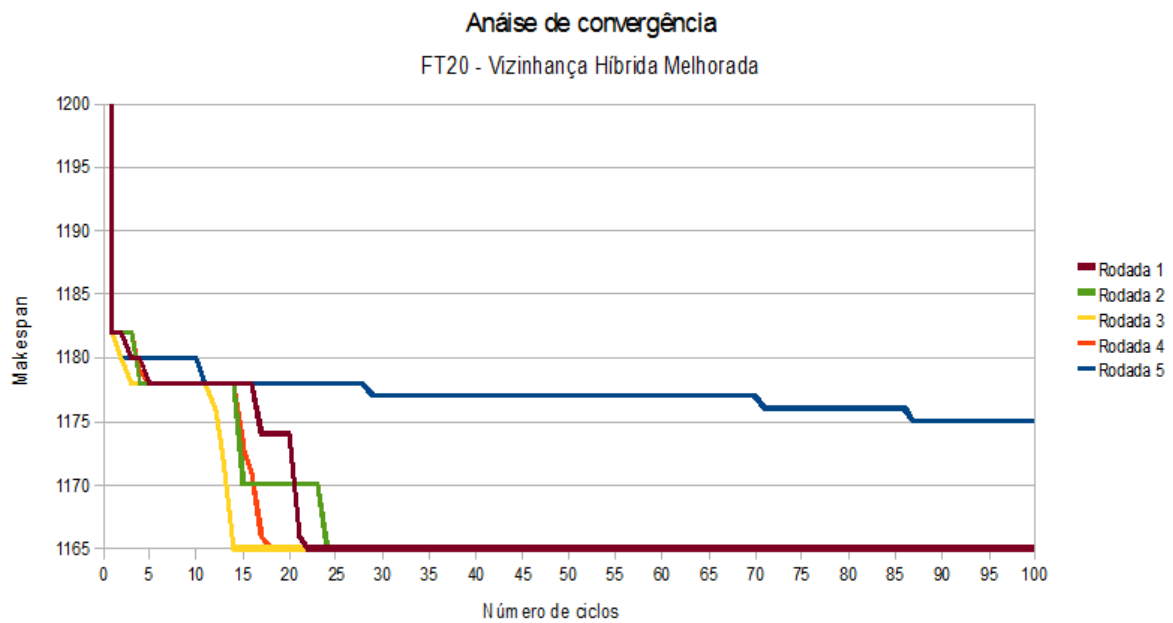


Figura 31 – Análise de convergência do *makespan* para o problema FT20 e o modelo de hibridização Vizinhança Híbrida Melhorada com a ordem dos operadores de geração de vizinhança Busca Tabu – Otimização por Multidão Similar de Partículas.

Fonte: O autor, 2010.

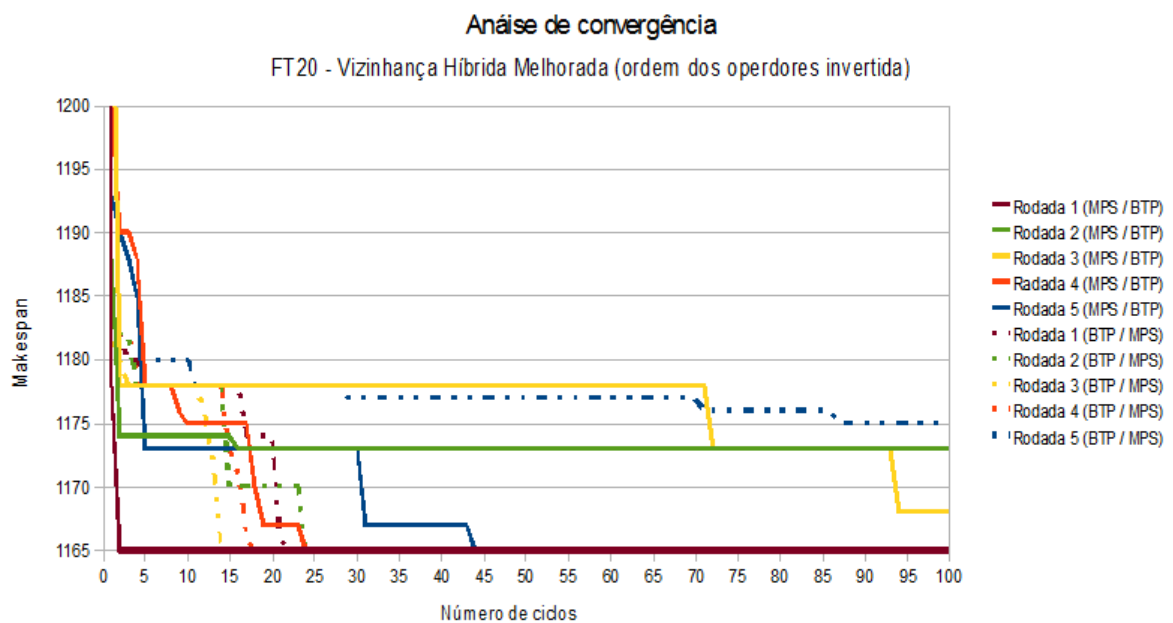


Figura 32 – Análise de convergência do *makespan* para o problema FT20 e o modelo de hibridização Vizinhança Híbrida Melhorada com a ordem dos operadores de geração de vizinhança Otimização por Multidão Similar de Partículas - Busca Tabu.

Fonte: O autor, 2010.

4.2.4 Resultados da análise de convergência para o modelo de hibridização Colisão de Partículas Similar

Ao contrário do que ocorre com os modelos anteriores, no Multi Colisão de Partículas Similar não existe interação entre as soluções e a convergência de cada solução de uma população inicial é independente da convergência das outras soluções, de forma que não se pode definir o ciclo no qual o algoritmo convergiu como um todo. Por este motivo nesta seção foi analisada a proporção de convergência em toda a população de soluções (ver Tab. 13).

Tabela 13 – Mínimo, máximo e média, em 12 soluções, do número de ciclos em que o modelo de hibridização Multi Colisão de Partículas Similar convergiu para a solução ótima do problema FT20. (NC = solução não convergiu antes de 100 ciclos - para calculo da média, quando a solução não converge para o ótimo, são considerados 100 ciclos).

		Número máximo de iterações sem melhora do <i>makespan</i> do operador Busca Tabu									
		100	200	300	400	500	600	700	800	900	1000
1 ^a rodada	min	3	1	1	1	1	1	1	1	1	1
	max	NC	NC	NC	NC	NC	NC	35	56	64	64
	med	91,92	81,33	40,92	33,83	53,17	23,08	14	10,92	12,58	12,58
2 ^a rodada	min	10	1	1	1	1	1	1	1	1	1
	max	NC	NC	NC	NC	NC	77	80	80	45	81
	med	89,08	57,08	45,08	30,58	28,17	14,67	25,67	10,33	23	18
3 ^a rodada	min	2	1	1	1	1	1	1	1	1	1
	max	100	NC	NC	NC	NC	NC	NC	60	64	96
	med	91,83	75,25	38,83	44,58	36,25	27,17	14,92	21,5	12,83	17,58
4 ^a rodada	min	NC	1	1	1	1	1	1	1	1	1
	max	NC	NC	NC	NC	NC	48	55	65	80	NC
	med	100	48,33	71,25	33,92	31,75	15,92	13,92	16,75	20,42	20,25
5 ^a rodada	min	NC	22	1	1	1	1	1	1	1	1
	max	NC	NC	NC	NC	91	85	NC	34	38	41
	med	100	88,42	81,17	25,92	20,17	23,5	26,17	9,08	9,67	15,83

Fonte: O autor, 2010.

Como base nesse fator de análise, foi observado que, quando o modelo de hibridização Multi Colisão de Partículas Similar foi aplicado ao problema FT20, fixando-se o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do operador Busca Tabu Paralela em 100, em 3 das 5 rodadas, apenas 1 ou 2 soluções convergiram para um ótimo global, sendo que nas 3 outras rodadas não houve conversão para um ótimo global de nenhuma das 12 soluções, dentro do limite de ciclos definido. Quando o valor desta

variável foi aumentado entre 100 e 600 houve um aumento significativo no número de soluções que convergiram para um ótimo global e uma redução na média do número de ciclos em que ocorre esta convergência, sendo que quando o valor desta variável foi fixado entre 600 e 1.000 não foi percebida nenhuma alteração significativa. Com base nestes resultados é possível concluir que o aumento do valor do parâmetro “número máximo de iterações sem melhora do *makespan*”, até um determinado limite, pode gerar uma melhora significativa da conversão do algoritmo.

4.2.5 Análise dos resultados encontrados com base no operador Busca Tabu Paralela

Quando os modelos de hibridização Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada são aplicados com os operadores de geração de vizinhança Busca Tabu Paralela e Otimização por Multidão de Partículas Similar, nesta ordem, e no caso em que o modelo de hibridização Vizinhança Híbrida é aplicado com a ordem dos operadores invertida, o operador Busca Tabu Paralela é inicialmente aplicado sobre o conjunto de soluções genitoras gerando os mesmos resultados que seriam gerados pela simples aplicação do método Busca Tabu à este mesmo conjunto de soluções e com os mesmo parâmetros. Posteriormente o Otimização por Multidão de Partículas Similar gera uma interação entre as soluções geradas pelo operador Busca Tabu Paralela ou, no caso do Vizinhança Híbrida, entre as mesmas soluções genitoras, dando origem a um grupo de soluções que formarão ou farão parte do novo grupo de soluções genitoras, sobre as quais será novamente aplicado o operador Busca Tabu Paralela. Desta forma espera-se que o *makespan* encontrado por qualquer um dos modelos aplicados seja menor, ou pelo menos igual, do que o menor *makespan* encontrado pela simples aplicação do método Busca Tabu a um dado conjunto de soluções. Já no caso em que a ordem dos operadores é invertida para os modelos de hibridização Aplicação Híbrida Sucessiva e Vizinhança Híbrida Melhorada, nenhuma afirmação pode ser feita uma vez que o operador Multidão de Partículas Similar altera o conjunto de soluções genitoras antes que o operador Busca Tabu Paralela seja aplicado pela primeira vez. O mesmo é válido para o Multi Colisão de Partículas Similar, uma vez que o operador de mutação M7 antecede a aplicação do operador Busca Tabu Paralela.

Conforme mencionado na seção 4.1 o aumento do valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu, até um determinado

ponto de saturação, gera uma melhora dos resultados do conjunto de soluções analisado de forma que espera-se que este aumento gere também uma melhora nos resultados encontrados pelos diferentes modelos de hibridização, independente da ordem dos operadores de geração de vizinhança. Conforme mostra a Fig. 33, quando o método Busca Tabu é aplicado às 12 soluções genitoras iniciais com o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” fixado em 700 uma das 12 soluções é convergida para uma solução ótima, de forma que os modelos de hibridização Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada com os operadores de geração de vizinhança Busca Tabu Paralela e Otimização por Multidão de Partículas Similar, nesta ordem, e Vizinhança Híbrida, com a ordem dos operadores invertida, são forçados a convergir em apenas 1 ciclos, o que pode ser verificado nas Tabs. 10, 11 e 12.

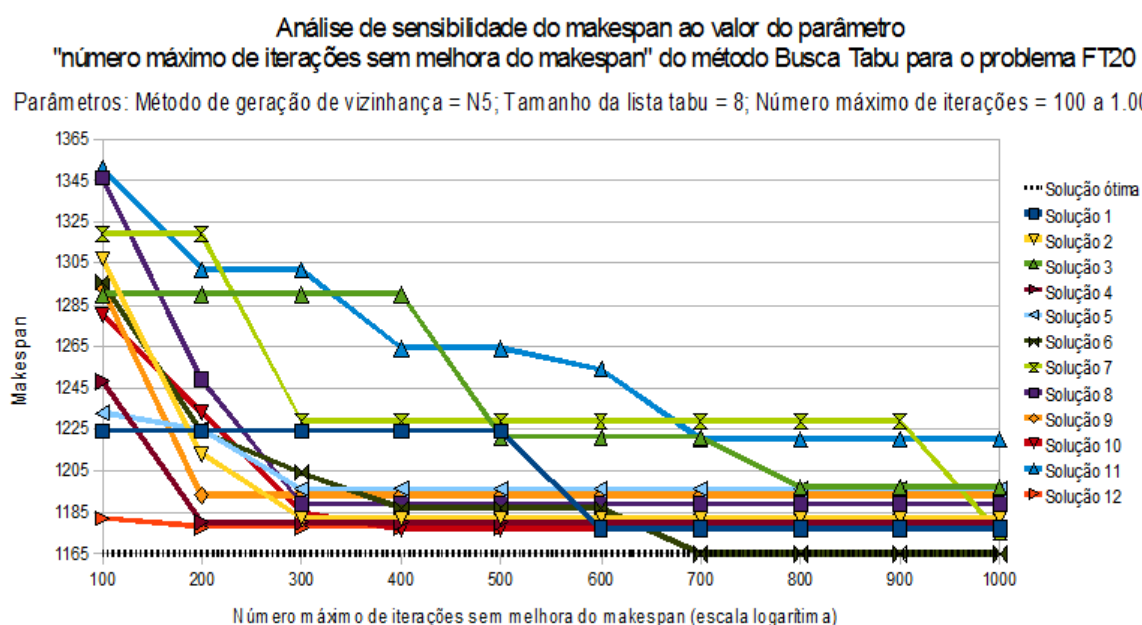


Figura 33 – Análise de sensibilidade do *makespan* à variação do parâmetro “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu para doze soluções iniciais geradas para o problema FT20.

Fonte: O autor, 2010.

4.2.6 Análise de convergência para os problemas FT10 e ABZ5 a ABZ9.

Quando os modelos de hibridização Aplicação Híbrida Sucessiva e Multidão de Partículas Similar foram aplicados aos problemas FT10, ABZ5 e ABZ6, foi verificado que os

algoritmos convergiram para a solução ótima em todas as rodadas sendo que esta conversão foi um pouco mais lenta para os problemas ABZ5 e ABZ6. No caso dos problemas ABZ7, ABZ8 e ABZ9, o teste de convergência extrapolou o tempo máximo de processamento estabelecido (1 semana) sem nem mesmo terminar os resultados da primeira rodada. Esses resultados refletem a complexidade que existe na solução do JSSP discutida na introdução deste trabalho.

Tabela 14 – Mínimo, máximo e média, em 5 rodadas, do número de ciclos em que o modelo de hibridização Aplicação Híbrida Sucessiva convergiu para a solução ótima dos problemas FT10, ABZ5, ABZ6, ABZ7, ABZ8 e ABZ9. (NC = solução não convergiu antes de 1.000 ciclos - para cálculo da média, quando a solução não converge para o ótimo, são considerados 1.000 ciclos).

		Problema					
		FT10	ABZ5	ABZ6	ABZ7	ABZ8	ABZ9
AHS BTP / MPS	min	4	13	27	NC	NC	NC
	max	21	100	172	NC	NC	NC
	med	8,8	60,2	68	NC	NC	NC

Fonte, O autor, 2010.

Tabela 15 – Mínimo, máximo e média, em 12 soluções, do número de ciclos em que o modelo de hibridização Multi Colisão de Partículas Similar convergiu para a solução ótima dos problemas FT10, ABZ5, ABZ6, ABZ7, ABZ8 e ABZ9. (NC = solução não convergiu antes de 1.000 ciclos - para cálculo da média, quando a solução não converge para o ótimo, são considerados 1.000 ciclos). * valor encontrado = 934 (ótimo = 930); ** valor encontrado = 1236 (ótimo = 1234); * valor encontrado = 945 (ótimo = 943).**

		Número máximo de iterações sem melhora do <i>makespan</i> do operador Busca Tabu					
		FT10	ABZ5	ABZ6	ABZ7	ABZ8	ABZ9
1ª rodada	min	1	1	1	NC	NC	NC
	max	231	NC**	653	NC	NC	NC
	med	73,33	600,67	218,75	NC	NC	NC
2ª rodada	min	1	1	1	NC	NC	NC
	max	NC*	NC**	NC***	NC	NC	NC
	med	221	626,17	370,08	NC	NC	NC
3ª rodada	min	2	67	1	NC	NC	NC
	max	247	NC**	NC***	NC	NC	NC
	med	75,58	851,83	259,33	NC	NC	NC
4ª rodada	min	1	1	72	NC	NC	NC
	max	247	NC**	NC***	NC	NC	NC
	med	82,92	778,92	456,5	NC	NC	NC
5ª rodada	min	1	1	1	NC	NC	NC
	max	219	NC**	826	NC	NC	NC
	med	78,25	664,33	307,92	NC	NC	NC

Fonte: O autor, 2010.

Como os testes avaliados foram aplicados à um reduzido conjunto de soluções iniciais e como foi observado pelo autor deste trabalho que o aumento desta variável gera uma aceleração considerável da velocidade de convergência do algoritmo, espera-se que o aumento do tamanho da população inicial facilite a convergência dos algoritmos propostos no caso dos problema maiores. Esta análise será realizada em trabalhos futuros.

4.3 **Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” do método Busca Tabu para os quatro modelos de hibridização.**

A figura 34, gerada com base nas Tabs. 35, 36 e 37, do anexo 2, apresenta uma comparação entre o menor *makespan* encontrado pela simples aplicação do método Busca Tabu às 12 soluções iniciais geradas para o problema FT10, e o melhor, a média e o pior dos valores dos resultados de 5 rodadas, gerados pela aplicação dos modelos de hibridização à este mesmo conjunto de soluções, para os valores do parâmetro “tamanho da lista tabu” do operador Busca Tabu Paralela ajustados entre 6 e 15. Nesta figura verifica-se que, independente do valor ajustado para o parâmetro “tamanho da lista tabu”, os valores encontrados para os modelos de hibridização Aplicação Híbrida Sucessiva (AHS), Vizinhança Híbrida (VH) e Vizinhança Híbrida Melhorada (VHM) flutuam entre o valor da solução ótima e o melhor valor encontrado pela simples aplicação do método Busca Tabu, conforme era esperado. Já no caso do Multi Colisão de Partículas Similar (CPS), apesar de não existir nenhuma restrição com relação aos valores que podem ser encontrados, quase todas as soluções geradas foram melhores do que a melhor solução encontrada pela simples aplicação do método Busca Tabu, demonstrando a superioridade de todos os modelos desenvolvidos.

A figura 35, gerada com base nas Tabs. 38, 39 e 40, do anexo 2, apresenta esta mesma comparação para o caso em que a ordem dos operadores de geração de vizinhança dos modelos de hibridização Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada é invertida. De forma surpreendente, apesar de alguns poucos valores terem sido piores do que a melhor solução gerada pelo Busca Tabu, o resultado geral foi ainda melhor neste caso, principalmente para o modelo de hibridização Aplicação Híbrida Sucessiva, onde a maioria das soluções corresponderam a solução ótima do problema.

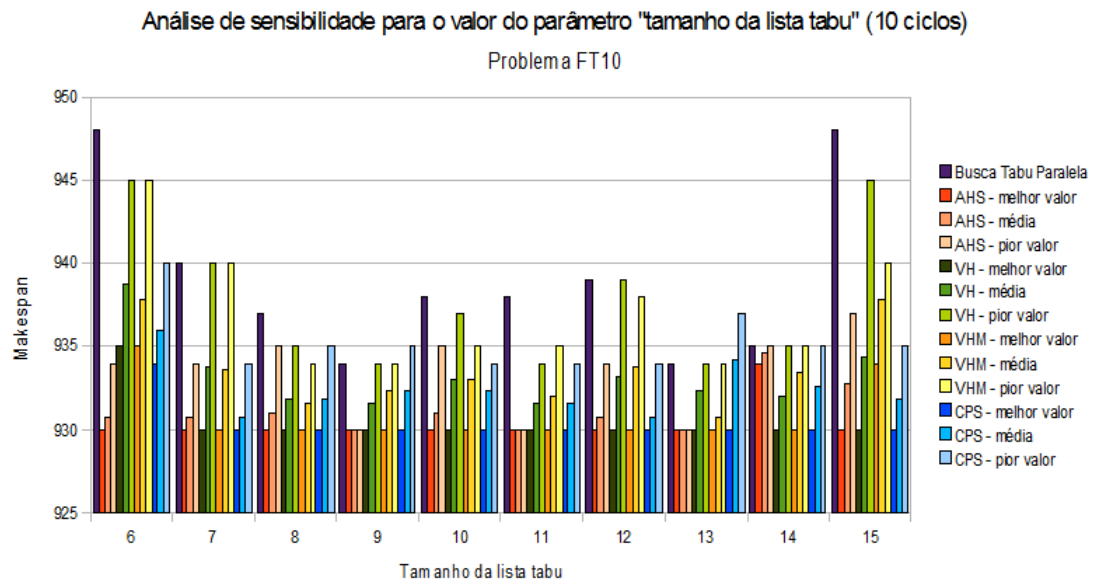


Figura 34 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” do método Busca Tabu para o problema FT10.

Fonte: O autor, 2010.

Análise de sensibilidade para o valor do parâmetro "tamanho da lista tabu" (10 ciclos) - Ordem inversa dos operadores

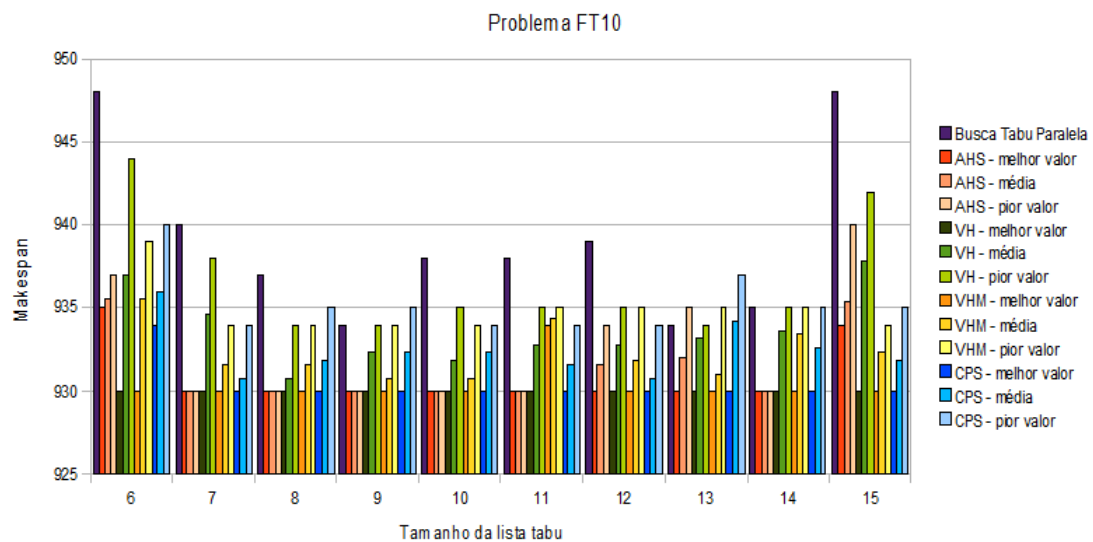


Figura 35 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” do método Busca Tabu para o problema FT10.

Fonte: O autor, 2010.

4.4 Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu para os quatro modelos de hibridização.

As Tabs. 41, 42, 43 e 44, do anexo 2, apresentam para doze soluções iniciais geradas para o problema FT10 análises da variação do *makespan*, quando o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu é ajustado entre 100, 1.000, 10.000 e 100.000, para 10 ciclos, e entre 10, 100, 1.000 e 10.000, para 100 ciclos para o Multi Colisão de Partículas Similar e para os modelos Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada, com os operadores de geração de vizinhança Busca Tabu Paralela e Multidão de Partículas Similar nesta ordem (BT/MPS), com 10 e 100 ciclos, e para o Multi Colisão de Partículas Similar e para os modelos Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada, com ordem invertida (MPS/BT), também com 10 e 100 ciclos. As Tabs. 55 a 58, 67 a 70, 73 a 76, 79 a 82, e 85 a 88, do anexo 2, apresentam estas mesmas análises respectivamente para os problemas ABZ5, ABZ6, ABZ7, ABZ8, e ABZ9.

Por estas tabelas é possível observar que, para os casos em que os modelos de hibridização foram aplicados aos problemas de tamanho 10 X 10 (FT10, ABZ5 e ABZ6), quase todos os resultados encontrados foram melhores do que a melhor solução gerada pelo método Busca Tabu quando o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do operador Busca Tabu Paralela foi ajustado a partir de 1.000, para 10 ciclos, e a partir de 100 para 100 ciclos, sendo que a partir destes valores o aumento do valor desta variável não resultou em uma melhora do *makespan*, mas gerou uma flutuação randômica entre o melhor valor encontrado pelo Busca Tabu e a solução ótima do problema. Para os caso em que os modelos de hibridização foram aplicados aos problemas de tamanho 20 X 15 (ABZ7, ABZ8 e ABZ9) os resultados encontrados foram melhores do que a melhor solução gerada pelo método Busca Tabu apenas quando o o valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do operador Busca Tabu Paralela foi ajustado a partir de 10.000 para 10 ciclos e 100 ciclos. Para todos os problemas testes os modelos de hibridização Multi Colisão de Partículas Similar e Aplicação Híbrida Sucessiva foram os que apresentaram os melhores resultados sendo que o Multi Colisão de Partículas Similar apresentou uma melhor convergência.

4.5 Análise de sensibilidade do *makespan* ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” do operador Otimização por Multidão de Partículas Similar para os modelos de hibridização Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada.

Para análise de sensibilidade do *makespan* à variação do valor parâmetro “porcentagem de partículas que sofrem mutação por iteração” do operador Otimização por Multidão de Partículas Similar dos modelos de hibridização Aplicação Híbrida Sucessiva (AHS), Vizinhança Híbrida (VH) e Vizinhança Híbrida Melhorada (VHM), foram testados os problemas FT10 e ABZ5, fixando-se o valor dos parâmetros “número máximo de iterações sem melhora do *makespan*” e “tamanho da lista tabu”, respectivamente, em 10.000 e 8, e o valor do parâmetro “número de ciclos” dos modelos de hibridização em 10, e variando-se o valor do parâmetro “porcentagem de partículas que sofrem mutação” entre 0 e 100. Desta análise verificou-se que, independente da ordem dos operadores de geração de vizinhança dos modelos de hibridização, todas as soluções encontradas apresentaram um *makespan* menor, ou pelo menos igual, ao da melhor solução gerada pelo Busca Tabu, quando aplicado às 12 soluções genitoras iniciais com os mesmos parâmetros fixos definidos para o operador Busca Tabu Paralela, sendo que no caso do problema FT10, a maioria das soluções encontradas foram soluções ótimas para o problema. Pode-se dizer que a flutuação dos valores encontrados entre o valor ótimo e a melhor solução encontrada pelo Busca Tabu é mais devida ao fator randômico que à variação do valor do parâmetro analisado. Dentre os modelos de hibridização analisados o Aplicação Híbrida Sucessiva, com os operadores Busca Tabu Paralela e Otimização por Multidão de Partículas Similar, nesta ordem, foi aquele que apresentou os melhores resultados e o comportamento mais uniforme. A inversão dos operadores aumentou a variância entre os resultados encontrados e gerou resultados melhores em alguns casos, principalmente para o problema ABZ5 onde não houve uma melhora das soluções geradas pelo Busca Tabu tão significativa quanto a que houve para o problema FT10.

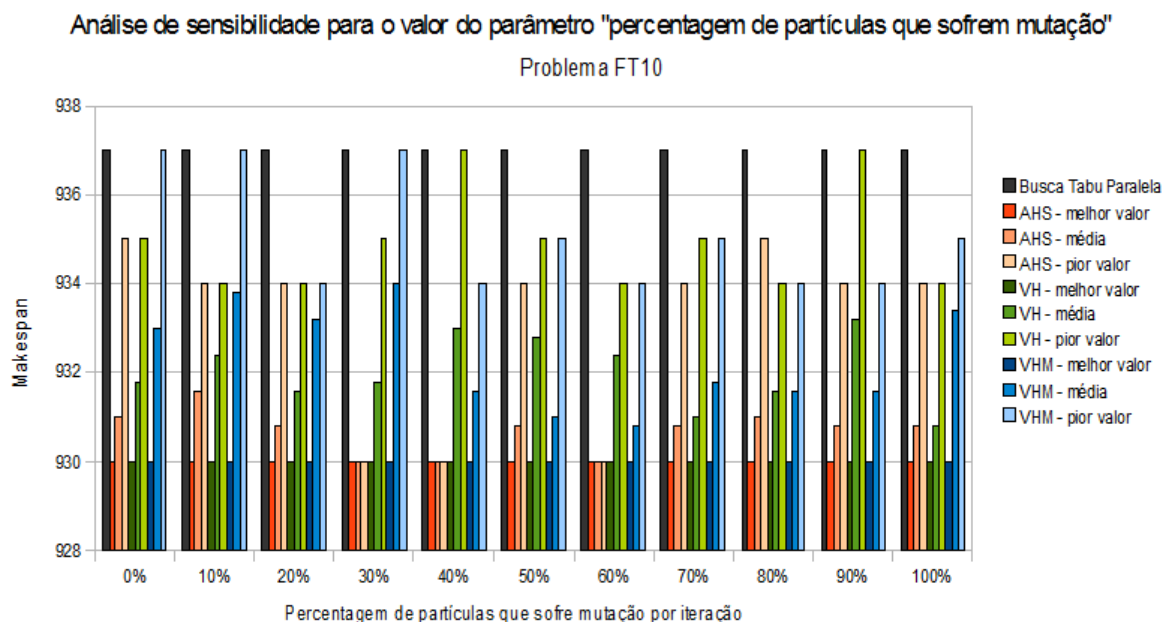


Figura 36 – a Análise de sensibilidade do *makespan* à variação do parâmetro “número de partículas que sofrem mutação” para os modelos de hibridização AHS, VH e VHM, com os operadores de geração de vizinhança Busca Tabu (BT) e Otimização por Multidão de Partículas Similar (OMPS), nesta ordem, para o problema FT10 (com os seguintes parâmetros fixos: número de ciclos = 10 – BT: método de geração de vizinhança = N5; e tamanho da lista tabu = 8; número máximo de iterações sem melhora do makespan = 10.000; – OMPS: método de cruzamento = C1 com 1 ponto de corte flutuante; método de mutação = M7).

Fonte: O autor, 2010.

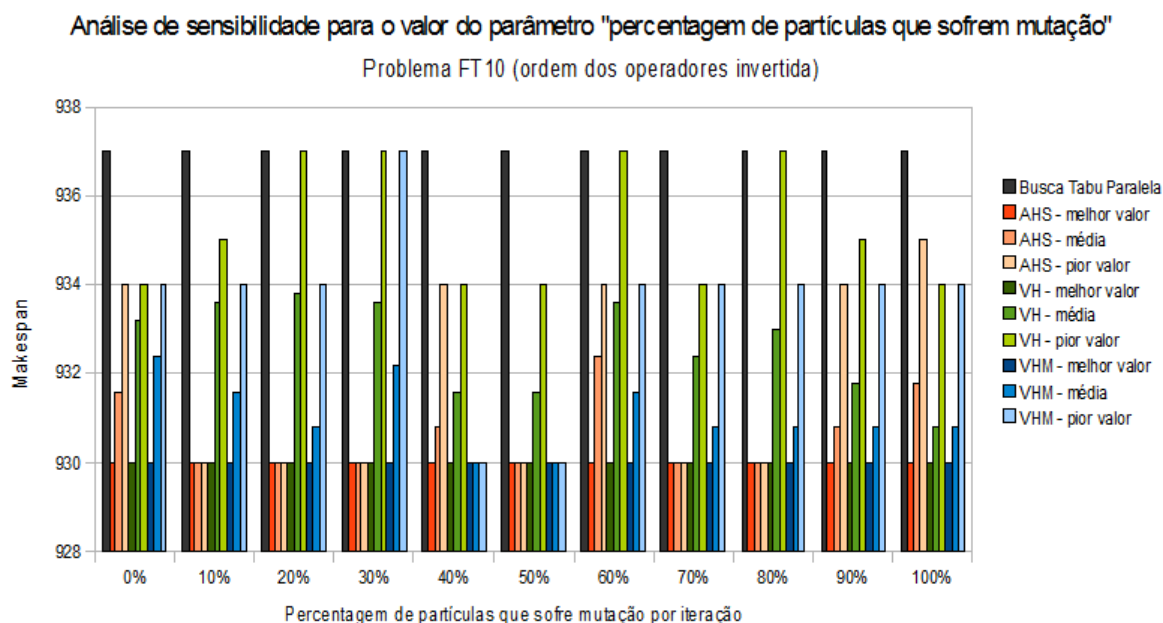


Figura 37 – Análise de sensibilidade do *makespan* à variação do parâmetro “número de partículas que sofrem mutação” para os modelos de hibridização AHS, VH e VHM, com os operadores de geração de vizinhança Busca Tabu (BT) e Otimização por Multidão de Partículas Similar (OMPS), com a ordem invertida, para o problema FT10 (com os seguintes parâmetros fixos: número de ciclos = 10 – BT: método de geração de vizinhança = N5; e tamanho da lista tabu = 8; número máximo de iterações sem melhora do makespan = 10.000; – OMPS: método de cruzamento = C1 com 1 ponto de corte flutuante; método de mutação = M7).

Fonte: O autor, 2010.

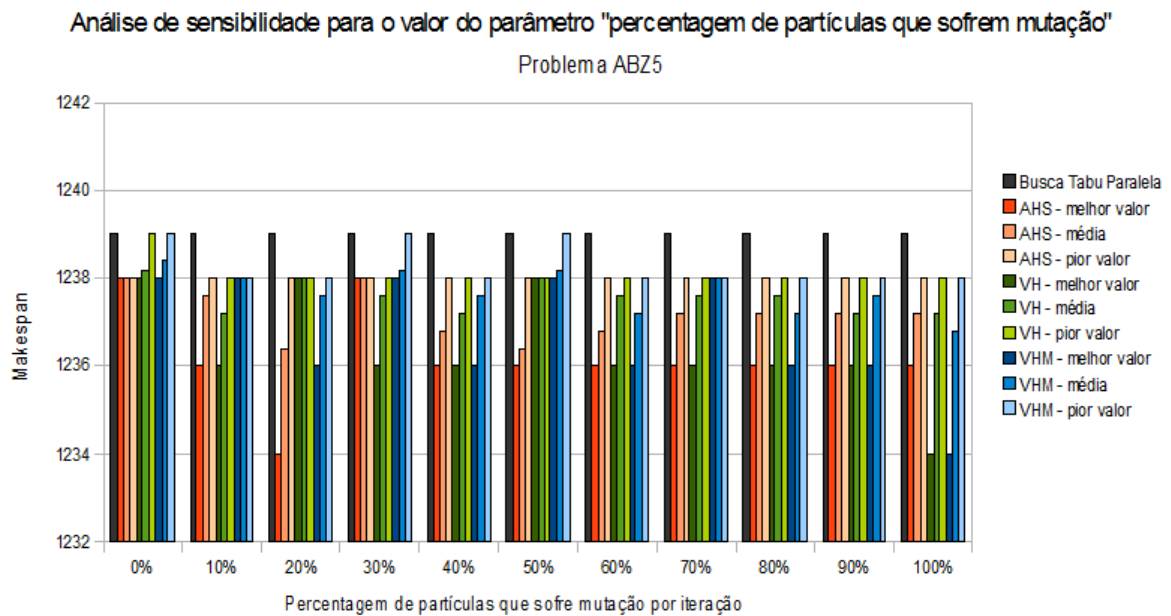


Figura 38 – a Análise de sensibilidade do *makespan* à variação do parâmetro “número de partículas que sofrem mutação” para os modelos de hibridização AHS, VH e VHM, com os operadores de geração de vizinhança Busca Tabu (BT) e Otimização por Multidão de Partículas Similar (OMPS), nesta ordem, para o problema ABZ5 (com os seguintes parâmetros fixos: número de ciclos = 10 – BT: método de geração de vizinhança = N5; e tamanho da lista tabu = 8; número máximo de iterações sem melhora do makespan = 10.000; – OMPS: método de cruzamento = C1 com 1 ponto de corte flutuante; método de mutação = M7).

Fonte: O autor, 2010.

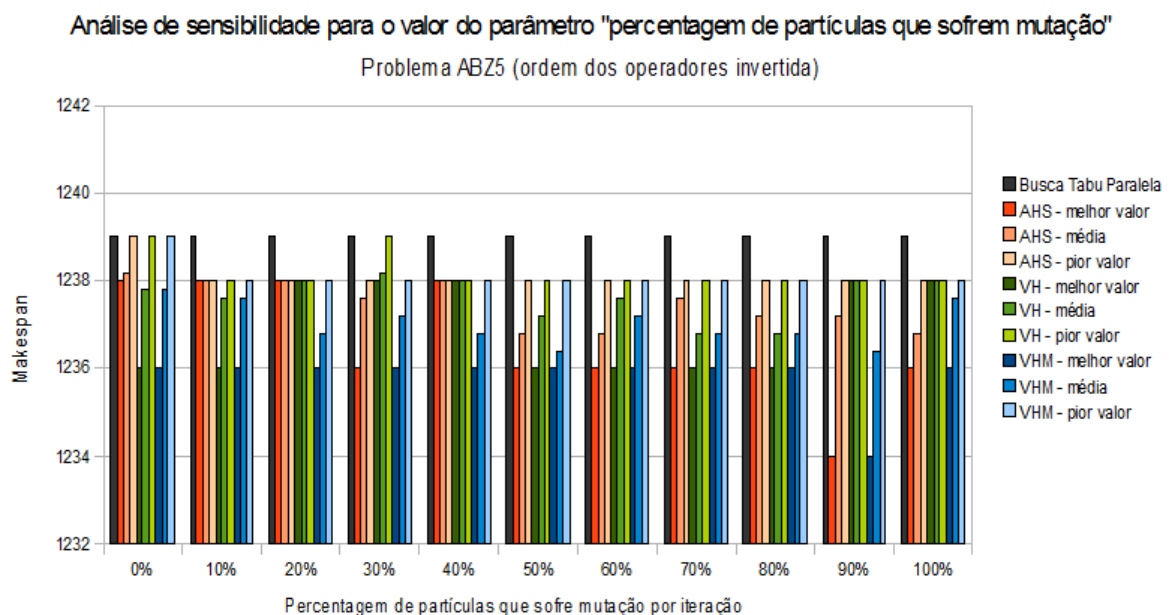


Figura 39 – Análise de sensibilidade do *makespan* à variação do parâmetro “número de partículas que sofrem mutação” para os modelos de hibridização AHS, VH e VHM, com os operadores de geração de vizinhança Busca Tabu (BT) e Otimização por Multidão de Partículas Similar (OMPS), com a ordem invertida, para o problema ABZ5 (com os seguintes parâmetros fixos: número de ciclos = 10 – BT: método de geração de vizinhança = N5; e tamanho da lista tabu = 8; número máximo de iterações sem melhora do makespan = 10.000; – OMPS: método de cruzamento = C1 com 1 ponto de corte flutuante; método de mutação = M7).

Fonte: O autor, 2010.

5 CONCLUSÕES E TRABALHOS FUTUROS

Na presente tese foram propostos, para solução do problema de escalonamento da produção em oficina de máquinas, quatro modelos de hibridização, sendo três deles – Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada – gerados com base no esquema geral utilizado para representação das Heurísticas de Busca Local e o quarto, Colisão de Partículas Similar, baseado no método Colisão de Partículas, até então aplicado apenas a problemas de natureza contínua. Apesar de, possivelmente, os três primeiros modelos já terem sido apresentados na literatura em contextos diversos e terem sido utilizados por outros autores, a análise de tais modelo, utilizando os algoritmos Otimização por Multidão de Partículas Similar proposto por Lian, Gu e Jiao (2006) e Busca Tabu, com uma versão determinística da estrutura de vizinhança apresentada por Nowicki e Smutnicki (1996) sem retro-propagação é, de acordo com a extensa pesquisa efetuada para elaboração deste trabalho, inovadora e pode contribuir de forma significativa para o meio científico. Adicionalmente, suas apresentações, conforme aqui sugeridas, podem constituir base para análise e desenvolvimento de inúmeros algoritmos híbridos, como os tratados nesta tese. Já o algoritmo desenvolvido para teste do Colisão de Partículas Similar, utilizando o método Busca Tabu, como operador de exploração local, e a técnica de mutação M7 apresentada por Lian, Gu e Jiao (2006), como operador de perturbação, desmonstrou através dos resultados obtidos nesta tese, que essa nova Heurística de Busca Local, a qual foi aqui concebida, provou ser mais robusta e eficiente que as outras Heurística de Busca Local até então conhecidas.

A seguir são listadas outras conclusões derivadas dos resultados obtidos:

- Todos os modelos de hibridização propostos foram capazes de gerar resultados melhores, ou pelo menos iguais, aos resultados gerados pela simples aplicação do método Busca Tabu ao mesmo conjunto de soluções iniciais e com os mesmos parâmetros fixos definidos para o operador Busca Tabu Paralela.
- No caso dos modelos de hibridização Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada com os operadores de geração de vizinhança Busca Tabu Paralela e Multidão de Partículas Similar, nesta ordem, e no caso do modelo de hibridização Vizinhança Híbrida com os mesmos operadores com a ordem invertida, o algoritmo é forçado a gerar resultados melhores, ou pelo menos iguais, aos resultados gerados pela simples aplicação do método Busca Tabu ao mesmo conjunto de soluções

iniciais e com os mesmos parâmetros fixos definidos para o operador Busca Tabu Paralela, independente dos valores definidos para os diferentes parâmetros.

- Os modelos de hibridização Aplicação Híbrida Sucessiva e Colisão de Partículas Similar foram os que apresentaram os melhores resultados em todos os testes analisados.
- Todos os algoritmos propostos apresentaram uma baixa sensibilidade tanto a variação do valor do parâmetro “tamanho da lista tabu” do operador Busca Tabu Paralela quanto a variação do valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” do operador Otimização por Multidão de Partículas Similar” (nos casos em que o mesmo se aplica).
- O tempo de processamento aumenta significativamente com o aumento do tamanho do problema (definido pelo número de máquinas e serviços).
- Apesar dos algoritmos não terem gerado soluções ótimas para os problemas ABZ7, ABZ8 e ABZ9 (todos com 15 máquinas e 20 serviços) os mesmo conseguiram encontrar soluções melhores do que as geradas pelo método Busca Tabu, sendo que acredita-se que o aumento do número de soluções iniciais pode acelerar o processo de convergência, de forma que as soluções ótimas possivelmente poderão ser geradas também para os problemas maiores aumentando-se o valor desta variável.

Na presente tese não foram exploradas todas as possibilidades de aplicação para os algoritmos propostos de forma que, em trabalhos futuros, pretende-se:

- Analisar como o tamanho da população interfere na convergência e nos resultados obtidos pelos algoritmos propostos;
- Analisar os algoritmos propostos com outros problemas testes apresentados na literatura;
- Testar os algoritmos propostos utilizando programação paralela.
- Testar os modelos de hibridização Aplicação Híbrida Sucessiva, Vizinhança Híbrida e Vizinhança Híbrida Melhorada com outros operadores de geração de vizinhança;
- Testar o modelo de hibridização Multi Colisão de Partículas Similar com outros operadores de exploração local e de perturbação.

REFERÊNCIAS

- ADAMS, J.; BALAS, E.; ZAWACK, D. The shifting bottleneck procedure for job-shop scheduling. **Management Science**, n. 34, pp. 391-401, 1988.
- AKERS, Sheldon B. A graphical approach to production scheduling problems. **Operations Research**, v. 4, n. 2, pp. 244–245, 1956.
- APPLEGATE, D.; COOK, W. A computational study of the job-shop scheduling instance. **ORSA Journal on Computing**, n. 3, pp. 149-156, 1991.
- AUTRIQUE, L.; SOUZA DE CURSI, J. E. On stochastic modification for global optimization problems: an efficient implementation for the control of the vulcanization process. **International Journal of Control**, v. 67, n. 1, pp. 1-22, 1997.
- AZIZI, N.; ZOLFAGHARI, S. Adaptive temperature control for simulated annealing: a comparative study. **Computers & Operations Research**, n. 31, pp. 2439-2451, 2004.
- BARNES, J. W.; CHAMBERS, J. B. Solving the job shop scheduling problem using tabu search. **IIE Transactions**, n. 27, pp. 257-263, 1995.
- BLAZEWICZ, J.; DOMSCHKE, W.; PESCH, E. The job shop scheduling problem: Conventional and new solution techniques. **European Journal of Operational Research**, n. 93, pp. 1-33, 1996.
- BRUCKER, P.; JURISCH, B.; SIEVER, B. A branch and bound algorithm for job-shop scheduling problem. **Discrete Applied Mathematics**, n. 49, pp. 105-127, 1994.
- CARLIER, J.; PINSON, E. An algorithm for solving the job shop problem. **Management Science**, V. 35, n. 29, pp.164-176, 1989.
- CHAMBERS, J. B.; BARNES, J. W. **New tabu search results for the job shop scheduling problem**. Technical report ORP96-10, Graduate Program in Operations Research and Industrial Engineering, The University of Texas at Austin, 1996.
- CHENG, Runwei; GEN, Mitsuo; TSUJIMURA, Yasuhiro. A tutorial survey of job-shop scheduling problems using genetic algorithms - I. representation. **Computers & Industrial Engineering**, v. 30, n. 4, pp. 983–997, 1996.
- CHENG, Runwei; GEN, Mitsuo; TSUJIMURA, Yasuhiro. A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies. **Computers & Industrial Engineering**, v. 36, n. 2, pp. 343–364, 1999.
- COLORNI, A.; DORIGO, M.; MANIEZZO, V.; TRUBIAN, M. Ant system for job-shop scheduling. **Belgian Journal of Operations Research**, n. 34, pp. 39-54, 1993.
- CONFEDERAÇÃO NACIONAL DA INDÚSTRIA. Indicadores de Competitividade na indústria brasileira : micro e pequenas empresas. CNI; SEBRAE. Brasília : CNI, 2006.
- DELL'AMICO, M.; TRUBIAN, M. Applying tabu-search to job-shop scheduling problem. **Annals of Operations Research**, 41(1–4), pp. 231–52, 1993.
- DEMIRKOL, Ebru; MEHTA, Sanjay; UZSOY, Reha. A computational study of shifting bottleneck procedures for shop scheduling problems. **Journal of Heuristics**, v. 3, n. 2, pp. 111–137, 1997.
- DORNDORF, U.; PESCH, E.; PHAN-HUY, T. Constraint propagation techniques for the

disjunctive scheduling problem. **Artificial Intelligence**, n. 122, pp. 189-240, 2000.

ES-SADEK, M. Z.; ELLAIA, R.; SOUZA DE CURSI, J. E. Application of anhybrid algorithm in a logistic problem. **Journal of Advanced Research in Applied Mathematics**, v. 1, pp. 34-52, 2009.

FIGLALI, Nilgün; ÖZKALE, Celal; ENGIN, Orhan; et al. Investigation of Ant System parameter interactions by using design of experiments for job-shop scheduling problems. **Computers & Industrial Engineering**, v. 56, n. 2, pp. 538–559, 2009.

FISHER, H.; THOMPSON, G. L. **Probabilistic learning combinations of local job-shop scheduling rules**. Muth, J. F., Thompson, G. L. (eds.). Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, 1963.

FOO, Simon Y.; TAKEFUJI, Yoshiyasu; SZU, Harold. Scaling properties of neural networks for job-shop scheduling. **Neurocomputing**, v. 8, n. 1, pp. 79–91, 1995.

FRAGA, T. B. **Desenvolvimento de uma ferramenta computacional para a programação da produção de empresas do setor de confecções do município de Nova Friburgo**. Dissertação de mestrado. Universidade do Estado do Rio de Janeiro, Instituto Politécnico, 2006.

FROMHERZ, M. P. J. **Constraint-based Scheduling**. Invited tutorial paper at the American Control Conference (ACC'01), Arlington, VA, June 2001. Disponível em :

<<http://www2.parc.com/isl/members/fromherz/publications/cbs-tutorial-local.pdf>>. Acesso em: 28/09/2009.

GLOVER, F. Tabu search–Part I. **ORSA Journal on Computing**, 1(3), pp. 190-206, 1989.

GLOVER, F. Tabu search–Part II. **ORSA Journal on Computing**, 2(1), pp. 4-32, 1990.

HEINONEN, J.; PETTERSSON, F. Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem. **Applied Mathematics and Computation**, v. 187, n. 2, pp. 989–998, 2007.

JACKSON, J. R. An extension of Johnson's result on job lot scheduling. **Naval Research Logistics Quarterly**, vol. 3, no. 3, pp. 201–203, 1956.

JAIN, A. S.; MEERAN, S. **A state-of-the-art review of job-shop scheduling techniques**.

Disponível em :

<ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia707_1s04/textos/jain98stateart.pdf>, Acesso em: 03/06/2008. 1998.

JAIN, A. S.; MEERAN, S. Deterministic job-shop scheduling: Past, present and future. **European Journal of Operations Research**, n. 113, pp. 390-434, 1999.

JOHNSON, S. M. Optimal two- and three-stage production schedules with set-up and times included. **Naval Research Logistics Quarterly**, n. 1, pp. 61-68, 1954.

KENNEDY, J.; EBERHART, R. **Particle Swarm Optimization**. IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, Perth, Australia, pp. 1942-1948, 1995.

KLEMMT, A.; HORN, S.; WEIGERT, G.; WOLTER, K. J. Simulation-based optimization vs. mathematical programming: A hybrid approach for optimizing scheduling problems. **Robotics and Computer-Integrated Manufacturing**, v. 25, Issue 6, December, pp. 917-925, 2009.

KNUPP, D.; SILVA NETO, A. J.; SACCO, W. F. Radiative properties estimation with the Particle Collision Algorithm based on a sensitivity analysis. **High Temperature - High Pressures**, v. 0, pp. 1 - 15, 2009.

- KOBAYASHI, S.; ONO, I.; YAMAMURA, M. **An efficient genetic algorithm for job shop scheduling problems**. In: Proceedings of ICGA'95, pp. 506-511, 1995.
- LAWER, E. L.; LENSTRA, J. K.; RINNOOY KAN, A. H. G.; SHMOYS, D. B. **Sequencing and Scheduling: Algorithms and Complexity**. Report BS-R89xx, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, 1989.
- LAWRENCE, S. **Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques** (Supplement). Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.
- LENSTRA, J. K.; RINNOOY KAN, A. H. G. **Towards a better algorithm for the jobshop scheduling problem** - 1. Rel. T  c., Stichting Mathematisch Centrum, 1973. Bibliographical data to be processed – Stichting Mathematisch Centrum. Mathematische Besliskunde; BN 22/73. ISSN: Stichting Mathematisch Centrum – 18 p.
- LIAN, Z.; GU, X.; JIAO, B. A similar particle swarm optimization algorithm for permutation flow shop scheduling to minimize makespan. **Applied Mathematics and Computation**, n. 175, pp. 773-785, 2006.
- MANNE, A. S. On the job-shop scheduling problem. **Operations Research**, v. 8, pp. 219-223, 1960.
- MATSUO, H.; SUH, C.J.; SULLIVAN, R. S. **A controlled search simulated annealing method for the general job-shop scheduling problem**. Working paper #03/04/88, Graduate School of Business, The University of Texas at Austin, Austin, Texas, USA, 1988.
- METROPOLIS, N.; ROSENBLUTH, M.; TELLER, A.; TELLER, E. Equation of state calculations by fast computing machines. **Journal Chemical Physics**, n. 21, pp. 1087-1092, 1953.
- NOWICKI, E.; SMUTNICKI, C. A fast taboo search algorithm for the job shop problem. **Management Science**, n. 42, pp. 797-813, 1996.
- ONO, I.; YAMAMURA, M.; KOBAYASHI, S. **A genetic algorithm for job-shop scheduling problems using job-based order crossover**. In: Proceedings of ICEC'96, pp. 547–552, 1996.
- PANWALKAR, S. S.; ISKANDER, W. A survey of scheduling rules. **Operations Research**, n   25, pp. 45-61, 1977.
- PEZZELLA, F.; MERELLI, E. A tabu search method guided by shifting bottleneck for the job shop scheduling problem. **European Journal of Operational Research** 120, pp. 297-310, 2000.
- SABUNCUOGLU, I.; GURGUN, B. A neural network model for scheduling problems. **European Journal of Operational Research** 93, pp. 288-299, 1996.
- SACCO, W. F.; OLIVEIRA, C. R. E. **A New Stochastic Optimization Algorithm based on Particle Collisions**. 2005 ANS Annual Meeting. Transactions of the American Nuclear Society 92, June 2005.
- SACCO, W. F.; OLIVEIRA C. R. E., PEREIRA C. M. N. A. Two stochastic optimization algorithms applied to nuclear reactor core design. **Progress in Nuclear Energy** 48, pp. 525 - 539, 2006.
- SHA, D. Y.; HSU, Cheng-Yu. A hybrid particle swarm optimization for job shop scheduling problem. **Computers & Industrial Engineering**, v. 51, n. 4, pp. 791–808, 2006.
- SHI, G. **A genetic algorithm applied to a classic job-shop scheduling problem**.

International Journal of Systems Science 28 (1), pp. 25–32, 1997.

SHI, Y.; EBERHART, R. C. **Parameter selection in particle swarm optimization.** In V.W. Porto, N. Saravanan, D. Waagen, A.E. Eiben (Eds.), Proceedings of the 7th international conference on evolutionary programming, pp. 591-600. New York: Springer, 1998

SOUZA DE CURSI, J. E. Informações fornecidas pelo autor. 2009.

SOUZA DE CURSI, J. E.; EL MOUATASIM, A.; ELLAIA, R. Random Perturbations of variable metric descent in unconstrained nonsmooth nonconvex perturbation. **International Journal of Applied Mathematics and Computer Science**, v. 16, pp. 463-474, 2006.

SOUZA DE CURSI, J. E.; POGU, M. Global optimization by random perturbation of the gradient method with a fixed parameter. **Journal of Global Optimization**, v. 5, pp. 159-180, 1994.

STENHÖFEL, K.; ALBRECHT, A.; WONG, C. K. An experimental analysis of local minima to improve neighbourhood search. **Computers & Operations Research** 30, pp. 2157–2173, 2003.

STORER, R., H.; WU, S. D.; VACCARI, R. New search spaces for sequencing instances with application to job shop scheduling. **Management Science** 38, pp. 1495-1509, 1992.

TAILLARD, É. D. Parallel taboo search technique for the jobshop scheduling problem. **ORSA Journal on Computing**, v.6, n. 2, Spring, 1994.

VAN LAARHOOVEN, P. J. M.; AARTS, E. H. L.; LENSTRA, J. K. Job shop scheduling by simulated annealing. **Operations Research**, v. 40, n. 1, pp. 113-125, 1992.

WANG, L.; ZHENG, D. An effective hybrid optimization strategy for job-shop scheduling problems. **Computers & Operations Research** 28, pp. 585-596, 2001.

WENQI, H.; AIHUA, Y. An improved shifting bottleneck procedure for the job shop scheduling problem. **Computers & Operations Research** 31, pp. 2093-2110, 2004.

WILLEMS, T. M.; ROODA, J. E. Neural networks for job-shop scheduling. **Control Engineering Practice**, v. 2, n. 1, pp. 31–39, 1994.

YAMADA, T.; NAKANO, R. **A genetic algorithm applicable to large-scale job-shop instances.** R. Manner, B. Manderick (eds.), Parallel instance solving from nature 2, North-Holland, Amsterdam, pp. 281-290, 1992.

YAMADA, T.; NAKANO, R. **Genetic algorithms for job-shop scheduling problems.** **Proceedings of Modern Heuristic for Decision Support**, pp. 67-81, UNICOM seminar, London, 18-19 March 1997.

ZHANG, C.; LI, P.; GUAN, Z.; RAO Y. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. **Computers & Operations Research** 34, pp. 3229 – 3242, 2007.

ZHANG, C.; LI, P.; RAO, Y.; GUAN, Z.. A very fast TS/SA algorithm for the job shop scheduling problem. **Computers & Operations Research** 3, pp. 282-294, 2008.

ZHANG, Defu; LI, Tangqiu; LI, Shaozi. **An improved shifting bottleneck procedure for the job shop scheduling problem.** In: The 9th International Conference on Computer Supported Cooperative Work in Design Proceedings. 2008, pp. 1112–1116.

APÊNDICE 1 – Problemas utilizados para testes

Tabela 16 – Problema FT6. UT = unidade de tempo; M = máquina.

Serviços	1ª máquina		2ª máquina		3ª máquina		4ª máquina		5ª máquina		6ª máquina	
	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT
S1	3	1	1	3	2	6	4	7	6	3	5	6
S2	2	8	3	5	5	10	6	10	1	10	4	4
S3	3	5	4	4	6	8	1	9	2	1	5	7
S4	2	5	1	5	3	5	4	3	5	8	6	9
S5	3	9	2	3	5	5	6	4	1	3	4	1
S6	2	3	4	3	6	9	1	10	5	4	3	1

Fonte: Fisher; Thompson, 1963.

Tabela 17 – Problema FT10. UT = unidade de tempo; M = máquina.

Serv.	1ª máquina		2ª máquina		3ª máquina		4ª máquina		5ª máquina		6ª máquina		7ª máquina		8ª máquina		9ª máquina		10ª máquina	
	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT
S1	1	29	2	78	3	9	4	36	5	49	6	11	7	62	8	56	9	44	10	21
S2	1	43	3	90	5	75	10	11	4	69	2	28	7	46	6	46	8	72	9	30
S3	2	91	1	85	4	39	3	74	9	90	6	10	8	12	7	89	10	45	5	33
S4	2	81	3	95	1	71	5	99	7	9	9	52	8	85	4	98	10	22	6	43
S5	3	14	1	6	2	22	6	61	4	26	5	69	9	21	8	49	10	72	7	53
S6	3	84	2	2	6	52	4	95	9	48	10	72	1	47	7	65	5	6	8	25
S7	2	46	1	37	4	61	3	13	7	32	6	21	10	32	9	89	8	30	5	55
S8	3	31	1	86	2	46	6	74	5	32	7	88	9	19	10	48	8	36	4	79
S9	1	76	2	69	4	76	6	51	3	85	10	11	7	40	8	89	5	26	9	74
S10	2	85	1	13	3	61	7	7	9	64	10	76	6	47	4	52	5	90	8	45

Fonte: Fisher; Thompson, 1963.

Tabela 18 – Problema FT20. UT = unidade de tempo; M = máquina.

Serviços	1ª máquina		2ª máquina		3ª máquina		4ª máquina		5ª máquina	
	M	UT	M	UT	M	UT	M	UT	M	UT
S1	1	29	2	9	3	49	4	62	5	44
S2	1	43	2	75	4	69	3	46	5	72
S3	2	91	1	39	3	90	5	12	4	45
S4	2	81	1	71	5	9	3	85	4	22
S5	3	14	2	22	1	26	4	21	5	72
S6	3	84	2	52	5	48	1	47	4	6
S7	2	46	1	61	3	32	4	32	5	30
S8	3	31	2	46	1	32	4	19	5	36
S9	1	76	4	76	3	85	2	40	5	26
S10	2	85	3	61	1	64	4	47	5	90
S11	2	78	4	36	1	11	5	56	3	21
S12	3	90	1	11	2	28	4	46	5	30
S13	1	85	3	74	2	10	4	89	5	33
S14	3	95	1	99	2	52	4	98	5	43
S15	1	6	2	61	5	69	3	49	4	53
S16	2	2	1	95	4	72	5	65	3	25
S17	1	37	3	13	2	21	4	89	5	55
S18	1	86	2	74	5	88	3	48	4	79
S19	2	69	3	51	1	11	4	89	5	74
S20	1	13	2	7	3	76	4	52	5	45

Fonte: Fisher; Thompson, 1963.

Tabela 19 – Problema ABZ5. UT = unidade de tempo; M = máquina.

Serv.	1ª máquina		2ª máquina		3ª máquina		4ª máquina		5ª máquina		6ª máquina		7ª máquina		8ª máquina		9ª máquina		10ª máquina	
	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT
S1	5	88	9	68	7	94	6	99	2	67	3	89	10	77	8	99	1	86	4	92
S2	6	72	4	50	7	69	5	75	3	94	9	66	1	92	2	82	8	94	10	63
S3	10	83	9	61	1	83	2	65	7	64	6	85	8	78	5	85	3	55	4	77
S4	8	94	3	68	2	61	5	99	4	54	7	75	6	66	1	76	10	63	9	67
S5	4	69	5	88	10	82	9	95	1	99	3	67	7	95	6	68	8	67	2	86
S6	2	99	5	81	6	64	7	66	9	80	3	80	8	69	10	62	4	79	1	88
S7	8	50	2	86	5	97	4	96	1	95	9	97	3	66	6	99	7	52	10	71
S8	5	98	7	73	4	82	3	51	2	71	6	94	8	85	1	62	9	95	10	79
S9	1	94	7	71	4	81	8	85	2	66	3	90	5	76	6	58	9	93	10	97
S10	4	50	1	59	2	82	9	67	8	56	10	96	7	58	5	81	6	59	3	96

Fonte: Adams, Balas; Zawack, 1988.

Tabela 20 – Problema ABZ6. UT = unidade de tempo; M = máquina.

Serv.	1ª máquina		2ª máquina		3ª máquina		4ª máquina		5ª máquina		6ª máquina		7ª máquina		8ª máquina		9ª máquina		10ª máquina	
	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT
S1	8	62	9	24	6	25	4	84	5	47	7	38	3	82	1	93	10	24	2	66
S2	6	47	3	97	9	92	10	22	2	93	5	29	8	56	4	80	1	78	7	67
S3	2	45	8	46	7	22	3	26	10	38	1	69	5	40	4	33	9	75	6	96
S4	5	85	9	76	6	68	10	88	4	36	7	75	3	56	2	35	1	77	8	85
S5	9	60	10	20	8	25	4	63	5	81	1	52	2	30	6	98	7	54	3	86
S6	4	87	10	73	6	51	3	95	5	65	2	86	7	22	9	58	1	80	8	65
S7	6	81	3	53	8	57	7	71	10	81	1	43	5	26	9	54	4	58	2	69
S8	5	20	7	86	6	21	9	79	10	62	3	34	1	27	2	81	8	30	4	46
S9	10	68	7	66	6	98	9	86	8	66	1	56	4	82	2	95	5	47	3	78
S10	1	30	4	50	8	34	3	58	2	77	6	34	9	84	5	40	10	46	7	44

Fonte: Adams, Balas; Zawack, 1988.

Tabela 21 – Problema ABZ7 UT = unidade de tempo; M = máquina.

Serv.	1 ^a máquina		2 ^a máquina		3 ^a máquina		4 ^a máquina		5 ^a máquina		6 ^a máquina		7 ^a máquina		8 ^a máquina		9 ^a máquina		10 ^a máquina		11 ^a máquina		12 ^a máquina		13 ^a máquina		14 ^a máquina		15 ^a máquina	
	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT
S1	3	24	4	12	10	17	5	27	1	21	7	25	9	27	8	26	2	30	6	31	12	18	15	16	14	39	11	19	13	26
S2	7	30	4	15	13	20	12	19	2	24	14	15	11	28	3	36	6	26	8	15	1	11	9	23	15	20	10	26	5	28
S3	7	35	1	22	14	23	8	32	3	20	4	12	13	19	11	23	10	17	2	14	6	16	12	29	9	16	5	22	15	22
S4	10	20	7	29	2	19	8	14	13	33	5	30	1	32	6	21	12	29	11	24	15	25	3	29	4	13	9	20	14	18
S5	12	23	14	20	2	28	7	32	8	16	6	18	9	24	10	23	4	24	11	34	3	24	1	24	15	28	13	15	5	18
S6	9	24	12	19	15	21	2	33	8	34	7	35	6	40	11	36	4	23	3	26	5	15	10	28	14	38	13	13	1	25
S7	14	27	4	30	7	21	9	19	13	12	5	27	3	39	10	13	15	12	6	36	11	21	12	17	2	29	1	17	8	33
S8	6	27	5	19	7	29	10	20	4	21	11	40	9	14	15	39	14	39	3	27	2	36	13	12	12	37	8	22	1	13
S9	14	32	12	29	9	24	4	27	6	40	5	21	10	26	1	27	15	27	7	16	3	21	11	13	8	28	13	28	2	32
S10	13	35	2	11	6	39	15	18	8	23	1	34	4	24	14	11	9	30	12	31	5	15	11	15	3	28	10	26	7	33
S11	11	28	6	37	13	29	2	31	8	25	9	13	15	14	5	20	4	27	10	25	14	31	12	14	7	25	3	39	1	36
S12	1	22	12	25	6	28	14	35	5	31	9	21	10	20	15	19	3	29	8	32	11	18	2	18	4	11	13	17	7	15
S13	13	39	6	32	3	36	9	14	4	28	14	37	1	38	7	20	8	19	12	12	15	22	2	36	5	15	10	32	11	16
S14	9	28	2	29	15	40	13	23	5	34	6	33	7	27	11	17	1	20	8	28	12	21	3	21	14	20	10	33	4	27
S15	10	21	15	34	4	30	13	38	1	11	12	16	3	14	6	14	2	34	9	33	5	23	14	40	11	12	7	23	8	27
S16	10	13	15	40	8	36	5	17	1	13	6	33	9	25	14	24	11	23	4	36	3	29	2	18	12	13	7	33	13	13
S17	4	25	6	15	3	28	13	40	8	39	2	31	9	35	7	31	12	36	5	12	11	33	15	19	10	16	14	27	1	21
S18	13	22	11	14	1	12	3	20	6	12	2	18	12	17	9	39	15	31	4	31	8	32	10	20	14	29	5	13	7	26
S19	6	18	11	30	8	38	15	22	14	15	12	20	10	16	4	17	2	12	3	13	13	40	7	17	9	30	5	38	1	13
S20	10	31	9	39	13	27	2	14	6	33	4	31	12	22	14	36	1	16	8	11	15	14	5	29	7	28	3	22	11	17

Fonte: Adams, Balas; Zawack, 1988.

Tabela 22 – Problema ABZ8. UT = unidade de tempo; M = máquina.

Serv.	1 ^a máquina		2 ^a máquina		3 ^a máquina		4 ^a máquina		5 ^a máquina		6 ^a máquina		7 ^a máquina		8 ^a máquina		9 ^a máquina		10 ^a máquina		11 ^a máquina		12 ^a máquina		13 ^a máquina		14 ^a máquina		15 ^a máquina	
	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT
S1	1	19	10	33	3	32	14	18	11	39	9	34	7	25	5	36	12	40	13	33	2	31	15	30	4	34	6	26	8	13
S2	10	11	11	22	15	19	6	12	5	25	7	38	1	29	8	39	14	19	12	22	2	23	4	20	3	40	13	19	9	26
S3	4	25	9	17	12	24	14	40	11	32	15	16	6	39	10	19	1	24	2	39	5	17	3	35	8	38	7	20	13	31
S4	15	22	4	36	3	34	13	17	5	30	14	12	2	13	7	25	10	12	8	18	11	31	1	39	6	40	9	26	12	37
S5	13	32	15	15	2	35	8	13	9	32	12	23	7	22	5	21	1	38	3	38	4	40	11	31	6	11	14	37	10	16
S6	11	23	13	38	9	11	15	27	10	11	7	25	6	14	5	12	3	27	12	26	8	29	4	28	14	21	1	20	2	30
S7	7	39	9	38	1	15	13	27	11	22	10	27	3	32	5	40	4	12	14	20	15	21	12	22	6	17	8	38	2	27
S8	12	11	14	24	11	38	9	15	10	19	15	13	6	30	1	26	3	29	7	33	13	21	2	15	4	21	5	28	8	33
S9	9	20	7	17	6	26	4	34	10	23	1	16	3	18	5	35	13	24	11	16	12	26	8	12	15	13	14	27	2	19
S10	2	18	8	37	15	27	10	40	6	40	7	17	9	22	4	17	11	30	1	38	5	21	13	32	12	24	14	24	3	30
S11	12	19	1	22	14	36	7	18	6	22	4	17	15	35	11	34	8	23	9	19	3	29	2	22	13	17	5	33	10	39
S12	7	32	4	22	13	24	6	13	5	13	2	11	1	11	14	25	9	13	3	15	11	33	12	17	15	16	10	38	8	24
S13	15	16	14	16	2	37	9	25	3	26	4	11	10	34	5	14	1	20	7	36	13	12	6	29	11	25	8	32	12	12
S14	9	20	11	24	12	27	10	38	6	34	13	39	8	33	5	37	3	31	14	15	15	34	4	33	7	26	2	36	1	14
S15	9	31	1	17	10	13	2	21	11	17	8	19	14	14	4	40	6	32	12	25	3	34	15	23	7	13	13	40	5	26
S16	9	38	13	17	4	14	14	17	5	12	2	35	7	35	1	19	11	36	8	19	10	29	3	31	6	26	12	35	15	37
S17	15	20	4	16	1	33	11	14	6	27	8	31	9	16	7	31	13	28	10	37	5	37	3	29	12	38	2	30	14	36
S18	12	18	4	37	15	16	7	15	9	14	13	11	14	32	6	12	2	11	11	29	8	19	5	12	10	18	3	26	1	39
S19	12	11	3	11	13	22	10	35	15	20	8	31	5	19	4	39	6	28	7	33	11	34	2	38	1	20	14	17	9	28
S20	3	12	13	25	6	23	9	21	7	27	10	30	15	23	12	39	4	26	14	34	8	17	2	24	5	12	1	19	11	36

Fonte: Adams, Balas; Zawack, 1988.

Tabela 23 – Problema ABZ9. UT = unidade de tempo; M = máquina.

Serv.	1 ^a máquina		2 ^a máquina		3 ^a máquina		4 ^a máquina		5 ^a máquina		6 ^a máquina		7 ^a máquina		8 ^a máquina		9 ^a máquina		10 ^a máquina		11 ^a máquina		12 ^a máquina		13 ^a máquina		14 ^a máquina		15 ^a máquina	
	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT	M	UT
S1	7	14	6	21	9	13	5	11	2	11	15	35	14	20	12	17	11	18	13	11	3	23	4	13	1	15	8	11	10	35
S2	2	35	6	31	1	13	4	26	7	14	10	17	8	38	13	20	11	19	14	12	9	16	5	34	12	15	15	12	3	14
S3	1	30	5	35	3	40	11	35	7	30	15	23	9	29	14	37	9	38	4	40	10	26	13	11	2	40	12	36	6	17
S4	8	40	6	18	5	12	9	23	1	23	10	14	14	16	13	14	11	23	4	12	7	16	15	32	2	40	12	25	3	29
S5	3	35	4	15	13	31	12	28	7	32	5	30	11	27	8	29	1	38	14	11	2	23	15	17	6	27	10	37	9	29
S6	6	33	4	33	7	19	13	40	11	19	1	33	14	26	3	31	12	28	8	36	5	38	2	21	15	25	10	40	9	35
S7	14	25	1	32	12	33	13	18	5	32	7	28	6	15	4	35	10	14	3	34	8	23	11	32	2	17	15	26	9	19
S8	3	16	13	33	10	34	12	30	14	40	9	12	15	26	6	26	7	15	4	21	2	40	5	32	1	14	8	30	11	35
S9	3	17	11	16	15	20	7	24	9	26	4	36	13	22	1	14	14	11	10	20	8	23	2	29	12	23	5	15	6	40
S10	5	27	10	37	4	40	12	14	14	25	8	30	1	34	3	11	6	15	13	32	2	36	11	12	15	28	9	31	7	23
S11	14	25	1	22	4	27	9	14	6	25	7	20	15	18	8	14	2	19	3	17	5	27	10	22	13	22	12	27	11	21
S12	15	34	11	15	1	22	4	29	14	34	7	40	8	17	3	32	13	20	6	39	5	31	12	16	2	37	9	33	10	13
S13	7	12	13	27	5	17	3	24	9	11	6	19	15	11	4	17	10	25	2	11	12	31	14	33	8	31	11	12	1	22
S14	6	22	15	15	1	16	9	32	8	20	5	22	10	11	14	19	2	30	13	33	7	29	12	18	4	34	11	32	3	18
S15	6	27	4	26	11	28	7	37	5	18	13	12	12	11	14	26	8	27	10	40	15	19	2	24	3	18	1	12	9	34
S16	9	15	6	28	10	25	7	32	2	13	8	38	12	11	3	34	5	25	1	20	11	32	4	23	13	14	15	16	14	20
S17	2	15	5	13	9	37	4	14	11	22	6	24	13	26	8	22	10	34	15	22	12	19	14	32	1	29	3	13	7	35
S18	8	36	6	33	14	28	10	20	11	30	5	33	15	29	1	34	4	22	12	12	7	30	9	12	2	35	3	13	13	35
S19	15	26	12	31	6	35	3	38	14	19	11	35	5	27	9	29	4	39	10	13	7	14	8	26	1	17	2	22	13	15
S20	2	36	8	34	12	33	9	17	15	38	7	39	6	17	4	27	14	29	3	16	1	16	5	19	10	40	13	35	11	39

Fonte: Adams, Balas; Zawack, 1988.

APÊNDICE 2 – Tabelas de resultados

Parâmetros

Tabela 24 – Parâmetros fixos definidos para aplicação do operador Busca Tabu.

Parâmetros 1		
Método: Busca tabu		
Método de geração de vizinhança	Tamanho da lista tabu	Número máximo de iterações sem melhora do <i>makespan</i>
N5	8	ver*

Fonte: O autor, 2010.

Tabela 25 – Parâmetros definidos para aplicação dos modelos de hibridização AHS, VH e VHM.

Parâmetros 2								
Tamanho da população	Número de ciclos	1° operador: Busca Tabu			2° operador: Multidão de Partículas Similar			
		Operador de geração de vizinhança	Tamanho da lista tabu	Número máximo de iterações	Operador de cruzamento	Operador de mutação	Porcentagem de mutação	Número de iterações
12	10 ou 100	N5	8	ver *	Rnd C1 (1pf)	M7	20	1

Fonte: O autor, 2010.

Tabela 26 – Parâmetros definidos para aplicação dos modelos de hibridização AHS, VH e VHM, com ordem dos operadores de geração de vizinhança invertida.

Parâmetros 3								
Tamanho da população	Número de ciclos	1° operador: Multidão de Partículas Similar				2° operador: Busca Tabu		
		Operador de cruzamento	Operador de mutação	Porcentagem de mutação	Número de iterações	Operador de geração de vizinhança	Tamanho da lista tabu	Número máximo de iterações
12	10 ou 100	Rnd C1 (1pf)	M7	20	1	N5	8	ver *

Fonte: O autor, 2010.

Tabela 27 – Parâmetros definidos para aplicação do modelo Colisão de Partículas Similar.

Parâmetros 4					
Tamanho da população	Número de iterações	Operador de Perturbação	Operador de Exploração Local: Busca tabu		
			Método de geração de vizinhança	Tamanho da lista tabu	Número máximo de iterações
12	10 ou 100	M7	N5	8	ver *

Fonte: O autor, 2010.

*1.000 (para FT6), 10.000 (para FT10, FT20, ABZ5e ABZ6) e 100.000 (para ABZ7, ABZ8 e ABZ9).

Problema: FT6 – Busca Tabu

Tabela 28 – a Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema FT6 (resultados obtidos pelo processador 2^o). Solução ótima para este problema: 55 UT.

Solução	Valor inicial	Número máximo de iterações sem melhora do <i>makespan</i>				
		100	1.000	10.000	100.000	1.000.000
1	71	55	55	55	55	55
2	98	55	55	55	55	55
3	102	55	55	55	55	55
4	114	55	55	55	55	55
5	89	55	55	55	55	55
6	89	55	55	55	55	55
7	90	55	55	55	55	55
8	97	55	55	55	55	55
9	100	55	55	55	55	55
10	75	55	55	55	55	55
11	82	55	55	55	55	55
12	80	55	55	55	55	55
Melhor solução	71	55	55	55	55	55
Nº total de soluções comput.	XXX	6593	41673	390601	3879836	38772145
Tempo total de CPU	XXX	1	4	39	381	3806

Fonte: O autor, 2010.

Tabela 29 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema FT6 (resultados obtidos pelo processador 2^o). Solução ótima para este problema: 55 UT.

Solução	Valor inicial	Tamanho da lista tabu									
		6	7	8	9	10	11	12	13	14	15
1	71	55	55	55	55	55	55	55	55	55	55
2	98	55	55	55	55	56	56	55	55	57	55
3	102	55	55	55	55	56	56	55	55	55	55
4	114	55	55	55	55	55	55	55	55	55	55
5	89	55	55	55	55	55	55	55	55	57	57
6	89	55	55	55	55	55	55	55	55	55	55
7	90	55	55	55	55	55	55	55	55	55	55
8	97	55	55	55	55	55	55	55	55	55	55
9	100	55	55	55	55	55	55	55	55	55	55
10	75	55	55	55	55	55	55	55	55	55	55
11	82	58	55	55	58	56	55	55	55	55	55
12	80	58	55	55	55	56	55	55	55	55	55
Melhor solução	71	55	55	55	55	55	55	55	55	55	55
Nº total de soluções computadas	XXX	41745	43320	41673	41801	43113	46678	43858	42982	44993	45174
Tempo total de CPU	XXX	4	4	4	4	5	5	4	5	5	5

Fonte: O autor, 2010.

^{**}Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Busca Tabu

Tabela 30 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema FT10 (resultados obtidos pelo processador 2^o). Solução ótima para este problema: 930 UT.

Solução	Valor inicial	Número máximo de iterações sem melhora do <i>makespan</i>				
		100	1.000	10.000	100.000	1.000.000
1	1668	1039	1025	990	990	990
2	1695	992	978	948	948	948
3	1977	1015	990	940	940	940
4	1807	980	945	945	945	945
5	1749	1077	945	944	944	944
6	1629	1064	976	949	949	949
7	1922	1027	954	940	940	940
8	1783	990	969	951	951	951
9	1701	1066	959	937	937	937
10	1766	1058	971	944	944	944
11	1763	1026	955	946	946	946
12	1640	1037	960	946	946	946
Melhor solução	1629	980	945	937	937	937
Nº total de soluções comput.	XXX	19188	170412	1188350	7371733	69188972
Tempo total de CPU	XXX	4	37	253	1559	14605

Fonte: O autor, 2010.

Tabela 31 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema FT10 (resultados obtidos pelo processador 2^o). Solução ótima para este problema 930 UT.

Solução	Valor inicial	Tamanho da lista tabu									
		6	7	8	9	10	11	12	13	14	15
1	1668	971	951	990	949	961	943	949	958	964	958
2	1695	948	945	948	945	943	954	951	951	935	952
3	1977	954	972	940	945	938	938	954	934	942	965
4	1807	980	963	945	934	953	956	952	955	950	958
5	1749	1031	940	944	945	946	950	955	946	951	961
6	1629	968	1041	949	966	960	951	951	945	955	967
7	1922	958	954	940	954	963	948	965	942	949	948
8	1783	956	953	951	951	940	940	947	952	951	955
9	1701	969	983	937	949	948	949	954	961	954	951
10	1766	1023	962	944	951	948	952	939	957	954	954
11	1763	969	940	946	958	973	948	948	956	966	956
12	1640	1013	1023	946	940	948	948	951	949	967	954
Melhor solução	1629	948	940	937	934	938	938	939	934	935	948
Nº total de soluções computadas	XXX	812764	970768	1188350	1078551	1144119	1375666	1175052	1198783	1187447	1165045
Tempo total de CPU	XXX	173	208	253	229	244	292	250	255	252	248

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Hibridização.

Tabela 32 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” (variando entre 6 e 9) com os parâmetros definidos nas Tabs. 25 e 27, exceto: número máximo de iterações sem melhora do *makespan* do operador Busca Tabu = 1.000; número de ciclos = 10; para o problema FT10 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 930 UT.

Problema	Número de ciclos	Tamanho da lista tabu	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	6	1	945	909910	193	930	907445	193	945	843338	181	934	826423	176
			2	938	860616	181	937	901539	191	940	911469	195	930	827295	174
			3	935	795575	167	930	878321	186	940	899904	195	940	835468	175
			4	940	846255	178	935	880897	187	939	932480	200	940	883284	186
			5	937	841776	178	934	894075	190	942	885021	188	945	859583	182
	10	7	1	938	1000763	210	934	936021	199	937	929965	199	937	902897	189
			2	938	907522	190	930	923627	195	940	924057	198	940	888532	186
			3	934	982192	206	935	899533	190	940	956727	205	940	876463	185
			4	934	843021	178	937	900723	191	934	941035	202	930	905605	191
			5	939	858925	181	934	955682	203	940	981937	210	937	904212	191
	10	8	1	930	959112	201	930	1020962	217	934	985962	211	937	913901	193
			2	934	919160	193	930	992560	210	945	917479	196	930	885483	186
			3	935	916335	194	930	1029824	218	937	918797	197	930	927191	197
			4	939	945029	199	938	1006946	213	930	977988	209	942	900220	189
			5	934	945899	199	930	962761	204	945	951110	204	934	899084	189
	10	9	1	937	981250	206	934	945529	201	930	958939	204	945	891639	188
			2	930	1048336	220	930	958027	203	935	1030413	221	934	905605	191
			3	945	868788	182	937	971235	206	940	973677	208	945	854502	180
			4	940	1012977	213	940	1035577	219	930	923016	197	945	915317	192
			5	943	988069	208	930	976790	207	942	978816	210	945	875486	184

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Híbridização.

Tabela 33 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” (variando entre 10 e 13) com os parâmetros definidos nas Tabs. 25 e 27, exceto: número máximo de iterações sem melhora do *makespan* do operador Busca Tabu = 1.000; número de ciclos = 10; para o problema FT10 (resultados obtidos pelo processador 2^o). Solução ótima para este problema: 930 UT.

Problema	Número de ciclos	Tamanho da lista tabu	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	10	1	935	936718	196	934	944883	201	944	1024766	219	939	924847	195
			2	938	960141	201	935	968338	206	945	950395	204	930	910206	192
			3	935	952090	200	934	949743	202	940	934769	199	940	872203	184
			4	935	1044925	220	937	933091	198	935	914994	197	940	937145	198
			5	940	1001981	211	940	1023652	217	940	983135	211	940	895951	189
	10	11	1	937	1006978	211	935	1000738	212	934	1019192	218	937	972191	205
			2	934	914085	192	939	1040041	220	939	1010929	216	945	935623	197
			3	939	996001	209	935	1022607	217	934	970238	208	943	940480	199
			4	930	971181	205	935	1052887	223	945	1024945	219	945	901276	189
			5	940	1019040	214	945	1011692	215	939	1040897	223	945	931334	196
	10	12	1	937	935674	197	935	965241	205	945	1017802	218	943	915611	194
			2	930	980739	206	934	955848	204	943	946932	203	945	902500	190
			3	945	974196	205	936	979307	208	945	954639	205	935	902707	191
			4	940	771625	163	939	941543	201	942	976702	210	945	948176	200
			5	938	974729	205	935	964059	205	945	957682	205	938	892012	189
	10	13	1	940	854407	180	935	994497	211	939	1011018	217	934	1000215	210
			2	935	979180	206	935	954944	203	944	1046603	225	946	967307	204
			3	935	862665	182	930	990827	211	944	1024382	220	938	914767	193
			4	938	934831	196	945	994084	212	949	1005883	216	937	940869	198
			5	948	937978	198	935	1031463	219	940	1036050	223	937	966605	203

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Hibridização.

Tabela 34 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” (variando entre 14 e 15) com os parâmetros definidos nas Tabs. 25 e 27, exceto: número máximo de iterações sem melhora do *makespan* do operador Busca Tabu = 1.000; número de ciclos = 10; para o problema FT10 (resultados obtidos pelo processador 2^o). Solução ótima para este problema: 930 UT.

Problema	Número de ciclos	Tamanho da lista tabu	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	14	1	942	1024739	216	936	965573	206	944	937875	201	944	878394	186
			2	938	1011521	213	941	971631	208	940	935312	202	948	892090	189
			3	946	1040044	220	944	914514	196	942	1006475	217	942	890655	188
			4	948	917042	194	944	916910	195	930	975814	209	944	929478	198
			5	937	914194	193	934	991011	211	948	967819	207	940	885339	187
	10	15	1	949	940094	198	936	965213	206	934	982073	212	934	922274	195
			2	940	931479	197	934	954958	204	954	960604	206	948	961788	204
			3	942	973918	206	940	1033402	219	945	991025	212	935	897019	189
			4	940	957519	203	938	965730	207	940	917058	197	941	913167	193
			5	940	912201	193	948	968133	206	949	967022	207	951	874357	185

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Hibridização.

Tabela 35 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” (variando entre 6 e 9) com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 930 UT.

Problema	Número de ciclos	Tamanho da lista tabu	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	6	1	935	6072615	1277	930	7134616	1514	935	6975502	1485	937	7106782	1518
			2	934	5638652	1184	930	7059489	1501	937	7101166	1509	935	6817226	1476
			3	940	5749370	1215	930	7285864	1534	937	6919793	1483	937	6829445	1475
			4	934	5435878	1141	930	6820756	1474	940	7413027	1559	935	7490158	1584
			5	937	5909775	1242	934	7128811	1516	945	6868513	1483	945	6633180	1461
	10	7	1	934	6401901	1345	930	7801066	1654	934	7511580	1594	934	7542778	1617
			2	930	6232552	1315	930	7852653	1656	940	7554822	1608	934	7281822	1574
			3	930	6227532	1297	930	7765125	1636	930	7547059	1607	940	7265884	1562
			4	930	6417864	1347	934	7601812	1617	935	7563000	1614	930	7447764	1596
			5	930	5912536	1239	930	7831678	1664	930	7613799	1622	930	7514072	1611
	10	8	1	930	7292540	1526	930	8273334	1759	930	8429162	1786	930	8175530	1747
			2	930	8090084	1692	930	8490570	1802	934	8456241	1794	934	7905123	1697
			3	930	7072673	1479	935	8749713	1857	935	8424164	1781	934	7928804	1691
			4	934	7695122	1604	930	8633197	1834	930	8254410	1748	930	8016407	1715
			5	935	7856376	1639	930	8959891	1903	930	8384108	1784	930	7703469	1645
	10	9	1	930	8330989	1745	930	8842913	1872	934	8307926	1751	930	7899382	1681
			2	934	9126795	1912	930	9108461	1928	930	8908318	1884	934	8260333	1756
			3	930	8321374	1740	930	8768483	1859	930	8366940	1767	930	8182443	1742
			4	934	8259361	1728	930	8766411	1854	934	8217366	1739	934	8228124	1749
			5	935	7766867	1623	930	8077135	1710	930	8851606	1867	934	8488801	1803

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Hibridização.

Tabela 36 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” (variando entre 10 e 13) com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 930 UT.

Problema	Número de ciclos	Tamanho da lista tabu	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	10	1	934	8617245	1807	930	8899738	1883	934	8484995	1792	935	8080598	1724
			2	930	9208571	1935	930	8941805	1895	930	8815811	1866	935	8793110	1879
			3	930	7204848	1510	930	8639943	1831	937	8391615	1775	935	7973271	1701
			4	934	8320399	1739	935	8428601	1786	934	9348423	1977	930	8592640	1837
			5	934	8402448	1756	930	9028551	1916	930	8906054	1883	930	8414205	1795
	10	11	1	934	8582415	1794	930	9373459	1987	930	8718764	1847	930	8662114	1847
			2	930	8161062	1707	930	9362256	1987	934	9232895	1954	935	8530027	1820
			3	934	8461672	1769	930	9310044	1974	930	9187435	1945	930	8986657	1916
			4	930	8847463	1852	930	9248454	1958	934	8895342	1879	935	8885255	1896
			5	930	7088955	1486	930	9008419	1906	930	8802577	1863	930	8481714	1808
	10	12	1	930	8249595	1730	930	9132865	1938	930	9031436	1912	930	8480397	1811
			2	930	8560952	1792	934	9282812	1970	930	9342613	1978	935	8706516	1860
			3	930	8897943	1860	930	8974023	1902	937	8895409	1884	938	8460398	1806
			4	934	8255265	1728	930	8773062	1861	930	8925576	1889	930	8719481	1860
			5	930	8270720	1731	930	8840939	1875	939	8751865	1855	936	8648872	1847
	10	13	1	937	7892420	1654	930	9258395	1967	934	8904441	1895	930	8347885	1784
			2	934	7889466	1663	930	8969121	1907	930	9692025	2056	930	8569581	1833
			3	935	7635717	1603	930	8853301	1883	930	8822965	1872	930	8273705	1770
			4	930	7892716	1656	930	8861735	1882	934	9276552	1971	934	8582575	1836
			5	935	7301961	1532	930	9467043	2010	934	9671393	2054	930	8876770	1898

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Hibridização.

Tabela 37 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” (variando entre 14 e 15) com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2^{}). Solução ótima para este problema: 930 UT.**

Problema	Número de ciclos	Tamanho da lista tabu	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	14	1	935	8638054	1818	934	9092212	1935	930	8690715	1847	934	8278627	1775
			2	930	8082139	1700	935	9223698	1965	930	8896324	1890	934	8422421	1802
			3	930	8449732	1773	935	9513086	2026	935	9200696	1956	930	8565769	1833
			4	934	9340572	1959	935	9163506	1949	935	9403061	1997	935	9338389	2001
			5	934	7670307	1612	934	8661222	1843	930	8913620	1893	934	8932295	1915
	10	15	1	934	8575188	1806	937	9048991	1931	930	8861125	1887	934	8482477	1820
			2	930	8641949	1816	930	9045743	1930	930	9075123	1930	935	8819824	1892
			3	935	8760342	1847	930	8734554	1862	930	8948169	1904	940	8772285	1881
			4	930	8233188	1732	937	9380261	2002	937	9142141	1946	940	8444151	1810
			5	930	8213942	1728	930	8549690	1826	945	8590396	1827	940	8378607	1799

Fonte: O autor, 2010.

^{**}Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Hibridização.

Tabela 38 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” (variando entre 6 e 9) com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 930 UT.

Problema	Número de ciclos	Tamanho da lista tabu	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	6	1	935	6072615	1277	936	6953976	1499	944	7023713	1507	937	7365564	1564
			2	934	5638652	1184	935	6941093	1487	937	7220775	1532	930	6989306	1494
			3	940	5749370	1215	935	7118147	1508	940	6778725	1454	938	6833400	1485
			4	934	5435878	1141	935	6823267	1477	934	6996935	1510	939	7438838	1582
			5	937	5909775	1242	937	7090006	1521	930	7246286	1536	934	7134775	1547
	10	7	1	934	6401901	1345	930	7622511	1615	934	7950459	1691	934	7388367	1587
			2	930	6232552	1315	930	7912625	1675	937	7744400	1659	930	7941791	1696
			3	930	6227532	1297	930	7641086	1621	938	7754891	1665	930	7732580	1660
			4	930	6417864	1347	930	7601367	1610	934	7880051	1661	930	7936702	1688
			5	930	5912536	1239	930	7916715	1681	930	7850304	1669	934	7766353	1657
	10	8	1	930	7292540	1526	930	8590662	1821	930	8536718	1810	934	8370929	1786
			2	930	8090084	1692	930	8279247	1757	930	8290998	1766	930	8842629	1888
			3	930	7072673	1479	930	8175983	1729	930	8303504	1766	934	8548164	1823
			4	934	7695122	1604	930	8736801	1852	930	8423540	1789	930	8715209	1861
			5	935	7856376	1639	930	8258241	1752	934	8392125	1780	930	8476801	1819
	10	9	1	930	8330989	1745	930	8674381	1839	934	8887860	1874	930	8462109	1803
			2	934	9126795	1912	930	9012290	1904	934	8863423	1876	934	9098836	1941
			3	930	8321374	1740	930	8641264	1828	934	8798393	1864	930	9199023	1955
			4	934	8259361	1728	930	8614168	1825	930	8195910	1731	930	9167338	1951
			5	935	7766867	1623	930	8309025	1761	930	8471717	1792	930	8407918	1793

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Híbridização.

Tabela 39 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” (variando entre 10 e 13) com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 930 UT.

Problema	Número de ciclos	Tamanho da lista tabu	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	10	1	934	8617245	1807	930	8724658	1851	930	8560667	1821	930	8884915	1895
			2	930	9208571	1935	930	8637506	1835	930	9084196	1929	930	8888109	1901
			3	930	7204848	1510	930	9023569	1915	935	9297431	1974	930	9649209	2059
			4	934	8320399	1739	930	8921280	1894	934	8928055	1896	930	9023013	1927
			5	934	8402448	1756	930	9121771	1938	930	8790236	1866	934	9558314	2038
	10	11	1	934	8582415	1794	930	9348633	1984	935	8987327	1906	934	9336228	1992
			2	930	8161062	1707	930	8641607	1837	930	8896843	1886	935	9511290	2026
			3	934	8461672	1769	930	9187466	1949	930	9209420	1951	934	9649352	2055
			4	930	8847463	1852	930	9230688	1959	935	8882791	1883	934	9228635	1968
			5	930	7088955	1486	930	8914959	1891	934	8524272	1808	935	9437254	2012
	10	12	1	930	8249595	1730	930	9686343	2057	930	8418371	1784	935	9694506	2068
			2	930	8560952	1792	934	9309073	1980	934	8483071	1799	930	9346168	1997
			3	930	8897943	1860	930	9488749	2016	930	9032137	1919	930	9113053	1945
			4	934	8255265	1728	934	9355737	1987	935	8963233	1905	930	9715976	2076
			5	930	8270720	1731	930	9051844	1921	935	9530649	2026	934	9589819	2046
	10	13	1	937	7892420	1654	930	9000858	1918	930	9469543	2018	930	9124419	1952
			2	934	7889466	1663	935	8834338	1882	934	8427207	1789	934	9327753	1996
			3	935	7635717	1603	935	9084664	1935	934	8920303	1898	938	9460475	2024
			4	930	7892716	1656	930	9717791	2068	934	9327224	1981	930	9549987	2044
			5	935	7301961	1532	930	9123433	1942	934	9069796	1930	930	9337634	2002

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Hibridização.

Tabela 40 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” (variando entre 14 e 15) com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2^{}). Solução ótima para este problema: 930 UT.**

Problema	Número de ciclos	Tamanho da lista tabu	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	14	1	935	8638054	1818	930	8721011	1860	934	9478867	2019	935	9734707	2090
			2	930	8082139	1700	930	9268148	1976	934	8952090	1908	930	9377953	2010
			3	930	8449732	1773	930	9522390	2028	930	9140148	1948	930	9772643	2094
			4	934	9340572	1959	930	10069443	2148	935	8535930	1820	930	9006814	1929
			5	934	7670307	1612	930	9334541	1988	935	9037379	1925	930	9891590	2121
	10	15	1	934	8575188	1806	940	9270763	1979	942	8956850	1915	934	10419813	2235
			2	930	8641949	1816	934	9680557	2067	930	8788319	1876	930	9345625	2009
			3	935	8760342	1847	935	8920039	1907	940	9314518	1988	934	9950065	2136
			4	930	8233188	1732	934	9515475	2033	939	8817291	1882	930	9515064	2045
			5	930	8213942	1728	934	9221390	1970	938	9287158	1983	934	9119335	1960

Fonte: O autor, 2010.

^{**}Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Hibridização.

Tabela 41 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do makespan” com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2^{}). Solução ótima para este problema: 930 UT.**

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	100	1	951	117873	26	953	103930	23	964	108706	24	940	96146	20
			2	936	133563	29	959	104561	22	954	100883	22	940	96390	21
			3	951	127979	27	945	115058	25	942	109774	24	949	95234	21
			4	960	125536	27	945	115791	25	960	107605	23	967	105055	23
			5	951	137937	30	951	112257	24	970	110415	24	963	104773	22
	10	1.000	1	935	955244	203	945	969534	205	943	936299	200	943	872188	186
			2	930	1017909	216	939	922632	195	938	1015331	219	937	903204	193
			3	940	1044232	222	944	931908	197	945	906902	195	945	880935	188
			4	940	951086	203	938	1034560	219	937	1038342	224	940	913110	195
			5	943	977044	209	930	1002012	213	944	960851	206	942	906226	193
	10	10.000	1	930	6610182	1410	930	8273334	1750	930	8429162	1806	930	8175530	1743
			2	930	7628456	1618	930	8490570	1793	934	8456241	1813	934	7905123	1691
			3	930	7099201	1515	935	8749713	1848	935	8424164	1804	934	7928804	1684
			4	935	8368694	1777	930	8633197	1826	930	8254410	1769	930	8016407	1711
			5	930	7009467	1492	930	8959891	1899	930	8384108	1804	930	7703469	1641
	10	100.000	1	930	51036059	10801	934	73308688	15452	930	68744252	14762	934	70546631	15081
			2	930	55549452	11775	934	69671902	14788	934	73364382	15578	930	71228404	15143
			3	930	55386167	11731	930	70929705	14982	930	69234545	14919	930	68568158	14687
			4	930	59120351	12540	930	71149085	14986	930	69725435	15036	930	69631880	14878
			5	934	57735192	12219	930	69990169	14787	934	68811749	14961	930	73582369	15565

Fonte: O autor, 2010.

^{**}Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Hibridização.

Tabela 42 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do makespan” com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 100, para o problema FT10 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 930 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	100	10	1	935	6072615	1277	930	7134616	1514	935	6975502	1485	937	7106782	1518
			2	934	5638652	1184	930	7059489	1501	937	7101166	1509	935	6817226	1476
			3	940	5749370	1215	930	7285864	1534	937	6919793	1483	937	6829445	1475
			4	934	5435878	1141	930	6820756	1474	940	7413027	1559	935	7490158	1584
			5	937	5909775	1242	934	7128811	1516	945	6868513	1483	945	6633180	1461
	100	100	1	934	6401901	1345	930	7801066	1654	934	7511580	1594	934	7542778	1617
			2	930	6232552	1315	930	7852653	1656	940	7554822	1608	934	7281822	1574
			3	930	6227532	1297	930	7765125	1636	930	7547059	1607	940	7265884	1562
			4	930	6417864	1347	934	7601812	1617	935	7563000	1614	930	7447764	1596
			5	930	5912536	1239	930	7831678	1664	930	7613799	1622	930	7514072	1611
	100	1.000	1	930	7292540	1526	930	8273334	1759	930	8429162	1786	930	8175530	1747
			2	930	8090084	1692	930	8490570	1802	934	8456241	1794	934	7905123	1697
			3	930	7072673	1479	935	8749713	1857	935	8424164	1781	934	7928804	1691
			4	934	7695122	1604	930	8633197	1834	930	8254410	1748	930	8016407	1715
			5	935	7856376	1639	930	8959891	1903	930	8384108	1784	930	7703469	1645
	100	10.000	1	930	8330989	1745	930	8842913	1872	934	8307926	1751	930	7899382	1681
			2	934	9126795	1912	930	9108461	1928	930	8908318	1884	934	8260333	1756
			3	930	8321374	1740	930	8768483	1859	930	8366940	1767	930	8182443	1742
			4	934	8259361	1728	930	8766411	1854	934	8217366	1739	934	8228124	1749
			5	935	7766867	1623	930	8077135	1710	930	8851606	1867	934	8488801	1803

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Hibridização.

Tabela 43 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do makespan” com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 930 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	100	1	951	117873	26	946	103869	23	958	116030	25	976	116094	25
			2	936	133563	29	951	111486	24	951	107662	23	955	107490	23
			3	951	127979	27	951	100398	21	940	107034	23	957	117155	26
			4	960	125536	27	951	110256	24	972	116540	25	946	140442	30
			5	951	137937	30	951	117156	25	946	126083	27	942	126355	27
	10	1.000	1	935	955244	203	935	943459	201	945	955677	202	943	968123	207
			2	930	1017909	216	945	972444	207	940	950843	202	938	1006514	216
			3	940	1044232	222	934	930513	198	945	1001965	214	937	1059413	227
			4	940	951086	203	944	1036839	220	945	976672	208	934	1025957	219
			5	943	977044	209	935	993555	210	944	977084	208	936	990789	212
	10	10.000	1	930	6610182	1410	930	8590662	1818	930	8536718	1805	934	8370929	1786
			2	930	7628456	1618	930	8279247	1758	930	8290998	1762	930	8842629	1889
			3	930	7099201	1515	930	8175983	1725	930	8303504	1761	934	8548164	1824
			4	935	8368694	1777	930	8736801	1849	930	8423540	1784	930	8715209	1864
			5	930	7009467	1492	930	8258241	1749	934	8392125	1776	930	8476801	1821
	10	100.000	1	930	51036059	10801	930	71372307	15140	937	72204555	15283	930	71509702	15242
			2	930	55549452	11775	930	70783000	15048	934	70415193	14991	930	70684082	15160
			3	930	55386167	11731	930	72838617	15399	934	69649071	14747	930	71689798	15248
			4	930	59120351	12540	930	70148427	14978	930	71060043	15078	934	71474379	15215
			5	934	57735192	12219	930	70535804	14973	934	70636564	14939	930	70525594	15146

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Hibridização.

Tabela 44 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do makespan” com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 100, para o problema FT10 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 930 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	100	10	1	935	6072615	1277	930	7134616	1514	935	6975502	1485	937	7106782	1518
			2	934	5638652	1184	930	7059489	1501	937	7101166	1509	935	6817226	1476
			3	940	5749370	1215	930	7285864	1534	937	6919793	1483	937	6829445	1475
			4	934	5435878	1141	930	6820756	1474	940	7413027	1559	935	7490158	1584
			5	937	5909775	1242	934	7128811	1516	945	6868513	1483	945	6633180	1461
	100	100	1	934	6401901	1345	930	7801066	1654	934	7511580	1594	934	7542778	1617
			2	930	6232552	1315	930	7852653	1656	940	7554822	1608	934	7281822	1574
			3	930	6227532	1297	930	7765125	1636	930	7547059	1607	940	7265884	1562
			4	930	6417864	1347	934	7601812	1617	935	7563000	1614	930	7447764	1596
			5	930	5912536	1239	930	7831678	1664	930	7613799	1622	930	7514072	1611
	100	1.000	1	930	7292540	1526	930	8273334	1759	930	8429162	1786	930	8175530	1747
			2	930	8090084	1692	930	8490570	1802	934	8456241	1794	934	7905123	1697
			3	930	7072673	1479	935	8749713	1857	935	8424164	1781	934	7928804	1691
			4	934	7695122	1604	930	8633197	1834	930	8254410	1748	930	8016407	1715
			5	935	7856376	1639	930	8959891	1903	930	8384108	1784	930	7703469	1645
	100	10.000	1	930	8330989	1745	930	8842913	1872	934	8307926	1751	930	7899382	1681
			2	934	9126795	1912	930	9108461	1928	930	8908318	1884	934	8260333	1756
			3	930	8321374	1740	930	8768483	1859	930	8366940	1767	930	8182443	1742
			4	934	8259361	1728	930	8766411	1854	934	8217366	1739	934	8228124	1749
			5	935	7766867	1623	930	8077135	1710	930	8851606	1867	934	8488801	1803

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Híbridização.

Tabela 45 – Análise de sensibilidade do *makespan* ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” (variando entre 0 e 30) com os parâmetros definidos nas Tabs. 25 e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2^{}). Solução ótima para este problema: 930 UT.**

Problema	Número de ciclos	% de partículas que sofrem mutação	Rodada	Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	0	1	930	7876621	1662	930	8208107	1720	930	7805406	1646
			2	935	8356488	1758	934	8152647	1719	934	7848526	1679
			3	930	8140021	1712	930	8100977	1695	934	8032535	1697
			4	930	8192694	1728	935	8106816	1702	930	7926658	1690
			5	930	8329256	1759	930	8283627	1729	937	7998262	1680
	10	10	1	930	8408978	1766	934	8305682	1741	934	7992729	1691
			2	934	8284742	1738	930	8036430	1685	937	7935137	1672
			3	934	8517219	1792	934	8442262	1769	930	7771456	1636
			4	930	8164518	1718	930	8416170	1764	934	8332943	1755
			5	930	8037462	1691	934	8072778	1715	934	8380661	1765
	10	20	1	930	8413556	1775	934	8185600	1717	934	8213737	1736
			2	930	8894962	1869	934	8132067	1695	934	8252281	1742
			3	930	8658018	1824	930	8660864	1813	934	8040867	1702
			4	934	8678071	1826	930	8263956	1728	930	8263413	1740
			5	930	8212016	1725	930	8177715	1712	934	8149949	1707
	10	30	1	930	8383984	1766	930	8267350	1736	934	8429185	1774
			2	930	8800579	1855	934	8741203	1824	930	8385066	1771
			3	930	8724938	1830	930	8131597	1702	935	7852261	1657
			4	930	8670066	1824	930	8677832	1814	937	7965481	1683
			5	930	8668371	1822	935	8442771	1775	934	8325577	1761

Fonte: O autor, 2010.

^{**}Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Híbridização.

Tabela 46 – Análise de sensibilidade do *makespan* ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” (variando entre 40 e 70) com os parâmetros definidos nas Tabs. 25 e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2^{*}). Solução ótima para este problema: 930 UT.**

Problema	Número de ciclos	% de partículas que sofrem mutação	Rodada	Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	40	1	930	9534918	2005	934	8124349	1716	934	8378110	1760
			2	930	8852427	1861	930	8870874	1851	930	8188046	1731
			3	930	9207326	1930	934	8324540	1753	930	7961143	1687
			4	930	8883452	1868	937	8129944	1709	930	7941385	1680
			5	930	8587727	1810	930	8549910	1805	934	8391970	1769
	10	50	1	930	8818791	1856	934	8907835	1868	930	8466188	1785
			2	930	8793525	1849	930	8389099	1755	930	8318324	1752
			3	930	8780911	1846	935	8685019	1816	930	8500050	1797
			4	934	8622737	1821	930	8557694	1794	935	8461337	1786
			5	930	9211649	1941	935	8631061	1804	930	8507578	1787
	10	60	1	930	8976031	1893	934	8670096	1817	930	8600095	1815
			2	930	9127711	1918	934	8421034	1771	930	8325065	1759
			3	930	9762423	2049	930	8581918	1799	930	8673254	1830
			4	930	9144546	1923	930	8809769	1846	934	8613099	1827
			5	930	9117720	1917	934	8701793	1830	930	8600890	1812
	10	70	1	934	9560284	2009	930	9030195	1912	934	8534885	1799
			2	930	9362965	1973	930	8711858	1820	930	8732332	1837
			3	930	9532805	2007	930	8685747	1826	930	8143112	1718
			4	930	9459716	1986	935	9035008	1890	930	8543229	1805
			5	930	9726195	2047	930	8743046	1833	935	8697472	1838

Fonte: O autor, 2010.

***Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Hibridização.

Tabela 47 – Análise de sensibilidade do *makespan* ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” (variando entre 80 e 100) com os parâmetros definidos nas Tabs. 25 e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2^{}). Solução ótima para este problema: 930 UT.**

Problema	Número de ciclos	% de partículas que sofrem mutação	Rodada	Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	80	1	930	9530915	2009	930	8509837	1783	930	8603316	1825
			2	935	9480159	1994	930	8447210	1762	930	8505105	1788
			3	930	9231650	1945	934	9057562	1902	934	9003796	1907
			4	930	9573708	2012	934	8986652	1879	930	8716395	1836
			5	930	9499618	1996	930	8515689	1789	934	8371492	1770
	10	90	1	930	9405960	1978	934	8978392	1872	930	8527549	1790
			2	930	9779329	2056	937	8927644	1868	934	8599363	1816
			3	934	9759936	2053	930	8855188	1849	930	8627786	1823
			4	930	9647505	2029	935	9254719	1943	934	9064165	1909
			5	930	9656846	2028	930	8964194	1881	930	8688131	1837
	10	100	1	930	9842059	2077	934	9335317	1951	934	8779875	1855
			2	930	9988789	2102	930	9381603	1967	930	8688794	1832
			3	930	9644348	2025	930	9195623	1924	935	8835544	1867
			4	930	9905147	2079	930	8989529	1876	934	8532656	1804
			5	934	10139366	2128	930	9071943	1903	934	8908747	1882

Fonte: O autor, 2010.

^{**}Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Híbridização.

Tabela 48 – Análise de sensibilidade do *makespan* ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” (variando entre 0 e 30) com os parâmetros definidos nas Tabs. 26 e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2^{}). Solução ótima para este problema: 930 UT.**

Problema	Número de ciclos	% de partículas que sofrem mutação	Rodada	Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	0	1	934	8298531	1743	934	7984253	1679	930	7665152	1618
			2	934	7684183	1628	934	7763541	1657	934	8184397	1731
			3	930	8153545	1717	934	8397742	1769	934	7801155	1645
			4	930	8013414	1681	930	8254549	1732	934	7861396	1655
			5	930	7911547	1668	934	7902683	1654	930	7266286	1536
	10	10	1	930	8094471	1700	934	8572434	1789	934	8207444	1734
			2	930	7736125	1621	934	8212600	1721	930	8065422	1703
			3	930	8163430	1710	935	8039763	1696	934	8096583	1713
			4	930	8427442	1764	935	8374251	1746	930	7960162	1673
			5	930	8507797	1788	930	7961478	1669	930	8217606	1729
	10	20	1	930	8487674	1777	934	8041480	1689	930	8329216	1759
			2	930	8411994	1772	937	8377697	1761	930	8318746	1766
			3	930	8252791	1733	934	8442494	1764	934	8272871	1757
			4	930	8421418	1768	930	8747571	1825	930	7954944	1681
			5	930	8299452	1746	934	8689494	1822	930	9129535	1917
	10	30	1	930	8471026	1778	937	8772163	1836	930	8775905	1854
			2	930	8816417	1858	937	7910400	1670	930	8443679	1784
			3	930	8280014	1738	930	8616066	1788	930	8779581	1849
			4	930	8512288	1788	930	8471653	1782	934	8711985	1831
			5	930	9055195	1912	934	8250145	1743	937	8553386	1825

Fonte: O autor, 2010.

^{**}Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Híbridização.

Tabela 49 – Análise de sensibilidade do *makespan* ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” (variando entre 40 e 70) com os parâmetros definidos nas Tabs. 26 e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2^{}). Solução ótima para este problema: 930 UT.**

Problema	Número de ciclos	% de partículas que sofrem mutação	Rodada	Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	40	1	934	9100518	1916	934	8509458	1784	930	8372597	1769
			2	930	8071011	1706	930	8496215	1776	930	8643138	1818
			3	930	8658359	1823	930	8755097	1836	930	9239153	1949
			4	930	8846834	1862	934	8768141	1843	930	9128505	1933
			5	930	8519857	1787	930	8694781	1816	930	9014562	1898
	10	50	1	930	8791724	1851	934	8797242	1839	930	9070800	1908
			2	930	8801748	1856	930	8442711	1763	930	9050594	1910
			3	930	8853747	1863	930	8713208	1829	930	9123464	1923
			4	930	8607462	1808	930	8572546	1803	930	8965730	1893
			5	930	8239288	1741	934	8894660	1859	930	9611243	2025
	10	60	1	934	9549691	2019	934	8439462	1772	934	9476796	1996
			2	934	9342532	1963	930	8859492	1860	930	8990201	1897
			3	934	9292789	1957	930	8628472	1801	930	9457756	2006
			4	930	9272407	1953	937	8385014	1756	930	9640933	2033
			5	930	9266614	1945	937	8855558	1866	934	9572296	2021
	10	70	1	930	8770795	1854	930	8770248	1833	930	9540894	2014
			2	930	9346959	1975	934	8675160	1816	930	9888038	2087
			3	930	9574041	2023	930	8779677	1837	930	9258499	1957
			4	930	9421440	1981	934	9105511	1908	934	9928144	2089
			5	930	9591680	2027	934	9006633	1882	930	9236172	1946

Fonte: O autor, 2010.

^{**}Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT10 – Modelos de Hibridização.

Tabela 50 – Análise de sensibilidade do *makespan* ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” (variando entre 80 e 100) com os parâmetros definidos nas Tabs. 26 e número de ciclos = 10, para o problema FT10 (resultados obtidos pelo processador 2^{}). Solução ótima para este problema: 930 UT.**

Problema	Número de ciclos	% de partículas que sofrem mutação	Rodada	Aplicação Híbrida Sucessiva			Vizinhaça Híbrida			Vizinhaça Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
FT10	10	80	1	930	9718439	2043	934	8824383	1849	930	8555977	1808
			2	930	9770717	2058	934	8955030	1875	930	9814187	2069
			3	930	9313543	1968	930	8796656	1844	934	9777857	2063
			4	930	9653832	2032	930	9048186	1889	930	9554215	2015
			5	930	9634286	2031	937	8566242	1799	930	10461386	2202
	10	90	1	930	9547193	2007	935	9291029	1939	930	10143555	2131
			2	930	9514393	2000	930	8913722	1869	930	10125984	2137
			3	934	9520370	1999	934	9480575	1983	930	10084114	2130
			4	930	9755453	2048	930	8699178	1816	934	10044125	2114
			5	930	9578920	2012	930	8624836	1812	930	9604210	2024
	10	100	1	935	9774134	2056	930	9241512	1931	930	9788610	2062
			2	934	9887416	2077	934	8624206	1803	934	9853937	2088
			3	930	9474682	1992	930	8844410	1853	930	9929253	2096
			4	930	9985311	2095	930	9053241	1893	930	10378507	2185
			5	930	10040783	2106	930	9194640	1923	930	10126267	2131

Fonte: O autor, 2010.

^{**}Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: FT20 – Busca Tabu.

Tabela 51 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema FT20 (resultados obtidos pelo processador 2^o). Solução ótima para este problema: 1165 UT.

Solução	Valor inicial	Número máximo de iterações sem melhora do <i>makespan</i>				
		100	1.000	10.000	100.000	1.000.000
1	2176	1224	1177	1177	1177	1177
2	2086	1307	1182	1175	1175	1175
3	2221	1290	1197	1197	1197	1197
4	2192	1248	1180	1178	1178	1178
5	2000	1233	1196	1196	1196	1196
6	2040	1296	1165	1165	1165	1165
7	2335	1319	1175	1175	1175	1175
8	2034	1346	1189	1189	1189	1189
9	2105	1293	1193	1193	1193	1193
10	2235	1280	1177	1177	1177	1177
11	2055	1351	1220	1219	1219	1219
12	2142	1182	1178	1178	1178	1178
Melhor solução	2000	1182	1165	1165	1165	1165
Nº total de soluções comput.	XXX	20634	130275	446831	3471672	33720178
Tempo total de CPU	XXX	5	28	101	802	7824

Fonte: O autor, 2010.

Tabela 52 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema FT20 (resultados obtidos pelo processador 2^o). Solução ótima para este problema 1165 UT.

Solução	Valor inicial	Tamanho da lista tabu									
		6	7	8	9	10	11	12	13	14	15
1	2176	1178	1178	1177	1178	1180	1177	1178	1177	1178	1178
2	2086	1199	1181	1175	1186	1185	1175	1182	1175	1176	1189
3	2221	1211	1178	1197	1178	1178	1178	1172	1178	1178	1173
4	2192	1215	1289	1178	1178	1174	1178	1173	1165	1192	1178
5	2000	1191	1175	1196	1188	1192	1170	1170	1176	1165	1165
6	2040	1199	1175	1165	1165	1165	1173	1175	1173	1173	1178
7	2335	1185	1178	1175	1190	1175	1185	1177	1177	1175	1177
8	2034	1182	1207	1189	1171	1175	1168	1165	1174	1178	1172
9	2105	1200	1191	1193	1178	1180	1180	1178	1188	1165	1178
10	2235	1198	1180	1177	1180	1180	1178	1182	1174	1227	1178
11	2055	1258	1199	1219	1165	1215	1178	1165	1175	1176	1178
12	2142	1182	1180	1178	1178	1182	1178	1180	1182	1180	1178
Melhor solução	2000	1178	1175	1165	1165	1165	1168	1165	1165	1165	1165
Nº total de soluções computadas	XXX	491126	497273	446831	441688	450066	425401	449002	426632	382174	347418
Tempo total de CPU	XXX	103	109	99	102	109	97	111	104	93	84

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ5 – Busca Tabu.

Tabela 53 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema ABZ5 (resultados obtidos pelo processador 2^o). Solução ótima para este problema: 1234 UT.

Solução	Valor inicial	Número máximo de iterações sem melhora do <i>makespan</i>				
		100	1.000	10.000	100.000	1.000.000
1	2177	1305	1248	1248	1239	1239
2	2030	1270	1265	1239	1239	1239
3	2373	1286	1249	1242	1242	1242
4	2286	1253	1239	1239	1239	1239
5	2739	1284	1249	1242	1242	1242
6	2180	1293	1239	1239	1239	1239
7	1928	1242	1242	1242	1242	1242
8	2086	1261	1248	1242	1242	1242
9	2190	1286	1249	1242	1242	1242
10	2025	1293	1239	1239	1239	1239
11	2181	1261	1255	1250	1250	1250
12	2211	1269	1251	1242	1242	1242
Melhor solução	1928	1242	1239	1239	1239	1239
Nº total de soluções comput.	XXX	20856	118263	773890	6287593	60760918
Tempo total de CPU	XXX	4	24	161	1332	12901

Fonte: O autor, 2010.

Tabela 54 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema ABZ5 (resultados obtidos pelo processador 2^o). Solução ótima para este problema 1234 UT.

Solução	Valor inicial	Tamanho da lista tabu									
		6	7	8	9	10	11	12	13	14	15
1	2177	1282	1238	1248	1248	1239	1239	1236	1238	1242	1245
2	2030	1239	1238	1239	1249	1245	1243	1244	1250	1238	1238
3	2373	1273	1243	1242	1243	1243	1242	1244	1242	1248	1245
4	2286	1250	1247	1239	1239	1243	1244	1243	1248	1242	1256
5	2739	1239	1250	1242	1236	1248	1239	1236	1238	1242	1242
6	2180	1264	1249	1239	1242	1243	1238	1242	1242	1238	1238
7	1928	1250	1249	1242	1242	1238	1248	1244	1248	1241	1239
8	2086	1242	1239	1242	1242	1243	1238	1236	1243	1251	1248
9	2190	1264	1250	1242	1236	1238	1239	1244	1247	1239	1243
10	2025	1279	1242	1239	1244	1239	1248	1249	1249	1248	1249
11	2181	1269	1238	1250	1239	1238	1240	1242	1242	1239	1236
12	2211	1238	1249	1242	1244	1249	1238	1244	1248	1246	1244
Melhor solução	1928	1238	1238	1239	1236	1238	1238	1236	1238	1238	1236
Nº total de soluções computadas	XXX	649783	683637	773890	989679	1040517	954236	1031281	994384	1162899	1175473
Tempo total de CPU	XXX	133	140	159	202	211	194	210	203	238	241

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ5 – Modelos de Híbridização.

Tabela 55 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 10, para o problema ABZ5 (resultados obtidos pelo processador 2 **). Solução ótima: 1234 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ5	10	100	1	1238	108726	22	1242	97652	20	1242	89155	19	1242	91966	19
			2	1238	102681	21	1242	95692	19	1242	94827	19	1239	87571	18
			3	1239	105711	22	1239	93331	20	1242	89737	19	1242	83228	17
			4	1241	106514	22	1242	91985	18	1242	91049	19	1242	88716	18
			5	1234	110297	22	1239	95727	20	1242	86764	17	1239	88061	19
	10	1.000	1	1236	804204	166	1238	782097	161	1238	735396	152	1238	697818	145
			2	1238	731900	151	1238	776398	159	1239	745587	154	1239	700248	145
			3	1238	719688	148	1238	746137	154	1239	713535	148	1238	702553	145
			4	1236	711408	146	1238	824691	169	1238	780644	160	1238	701868	146
			5	1238	834873	172	1238	761328	157	1238	717144	147	1236	703403	145
	10	10.000	1	1238	6015416	1252	1238	6990169	1464	1236	6912764	1448	1238	6724856	1408
			2	1236	5975992	1249	1238	7009377	1461	1238	6792853	1430	1239	6553340	1409
			3	1236	5946901	1246	1238	6865216	1434	1238	6909796	1442	1236	6662816	1398
			4	1236	6157942	1287	1236	7151718	1492	1238	6734397	1421	1238	6541822	1365
			5	1236	6103835	1276	1238	6960612	1447	1238	6956770	1450	1239	6661198	1399
	10	100.000	1	1236	45709932	9661	1236	62004981	13230	1238	62301427	13385	1238	61537351	13176
			2	1238	53933674	11443	1236	61761882	13154	1238	62558386	13446	1238	62101372	13547
			3	1236	47395863	10024	1238	62398139	13360	1238	61780974	12844	1238	61362930	13105
			4	1238	52533467	11067	1238	62221864	13454	1236	62081028	13135	1236	61775319	13118
			5	1238	45519799	9697	1238	62146763	13337	1238	61400978	12937	1238	62062621	13311

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ5 – Modelos de Híbridização.

Tabela 56 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do makespan” com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 100, para o problema ABZ5 (resultados obtidos pelo processador 2 **). Solução ótima: 1234 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ5	100	10	1	1251	98152	22	1244	96486	21	1242	90995	19	1239	82566	18
			2	1239	101071	21	1242	93037	20	1242	90978	20	1242	85579	18
			3	1239	100892	22	1250	90178	20	1249	94444	20	1252	77481	18
			4	1239	104991	22	1239	99397	21	1239	86713	19	1253	89351	19
			5	1242	103553	22	1239	101200	21	1244	84082	19	1245	78780	18
	100	100	1	1238	747123	155	1239	751857	156	1234	721674	149	1238	716792	150
			2	1238	769049	160	1239	770043	158	1239	716078	147	1238	695136	145
			3	1236	734372	154	1239	763103	158	1242	700157	146	1238	710893	146
			4	1234	742455	155	1234	768322	159	1239	705009	146	1238	719829	148
			5	1239	771693	159	1234	754994	157	1234	722395	150	1239	705501	147
	100	1.000	1	1236	7513915	1553	1236	7371602	1515	1236	6926417	1419	1236	6763453	1394
			2	1234	6846162	1415	1236	7354730	1506	1236	6957177	1426	1236	6862461	1419
			3	1236	7332730	1508	1236	7399881	1519	1238	6701279	1380	1236	6797030	1402
			4	1236	7321776	1513	1236	7264290	1494	1236	6729392	1378	1236	6798990	1403
			5	1236	7236223	1485	1236	7252580	1489	1236	6795200	1392	1236	6787292	1396
	100	10.000	1	1236	68052742	14152	1238	67533029	14057	1236	66802260	14010	1236	65508093	13801
			2	1236	68244373	14233	1236	68695487	14286	1236	65630166	13723	1234	65418746	13709
			3	1236	67151224	13980	1236	69042955	14390	1236	64668262	13427	1236	64281883	13477
			4	1236	69300690	14475	1236	69121515	14375	1236	64521186	13286	1236	65291997	13726
			5	1236	69830093	14599	1236	69512765	14501	1236	64559561	13362	1238	64992699	13470

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ5 – Modelos de Híbridização.

Tabela 57 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 10, para o problema ABZ5 (resultados obtidos pelo processador 2 **). Solução ótima: 1234 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ5	10	100	1	1238	108726	22	1242	84801	18	1239	98472	20	1242	81623	17
			2	1238	102681	21	1238	88190	18	1239	88676	19	1248	88365	19
			3	1239	105711	22	1238	92199	19	1234	91467	18	1242	94714	20
			4	1241	106514	22	1239	92910	19	1242	91983	19	1242	84876	17
			5	1234	110297	22	1234	84188	17	1239	90927	19	1239	93756	20
	10	1.000	1	1236	804204	166	1239	729118	150	1238	733702	151	1238	805274	166
			2	1238	731900	151	1236	762073	157	1238	792967	163	1238	802460	166
			3	1238	719688	148	1238	695821	142	1238	801435	165	1238	778230	161
			4	1236	711408	146	1234	781385	161	1239	758860	156	1238	770089	158
			5	1238	834873	172	1236	839493	172	1239	762392	157	1239	796702	165
	10	10.000	1	1238	6015416	1252	1238	6798623	1417	1238	6783910	1415	1238	7702244	1610
			2	1236	5975992	1249	1238	7177434	1501	1238	6975795	1452	1236	7478789	1566
			3	1236	5946901	1246	1238	7148265	1500	1238	6787266	1415	1236	7278149	1541
			4	1236	6157942	1287	1238	7060412	1480	1238	6889769	1442	1238	7496403	1556
			5	1236	6103835	1276	1236	7311356	1527	1238	6945281	1445	1238	7205433	1541
	10	100.000	1	1236	45709932	9661	1238	62494297	13199	1238	62717393	13576	1238	62567778	13506
			2	1238	53933674	11443	1236	62317500	13318	1238	62374289	13382	1234	62326416	13509
			3	1236	47395863	10024	1238	62284855	13284	1238	61891416	13178	1238	62156646	12962
			4	1238	52533467	11067	1238	62053307	13214	1238	62119263	13402	1236	61924018	13296
			5	1238	45519799	9697	1234	62553048	13425	1238	61745463	13118	1236	62101820	13142

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ5 – Modelos de Híbridização.

Tabela 58 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do makespan” com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 100, para o problema ABZ5 (resultados obtidos pelo processador 2 **). Solução ótima: 1234 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ5	100	10	1	1251	98152	22	1239	97364	21	1248	93564	20	1251	98152	22
			2	1239	101071	21	1244	93046	20	1249	89061	20	1239	101071	21
			3	1239	100892	22	1242	102719	22	1244	88637	18	1239	100892	22
			4	1239	104991	22	1242	101335	21	1258	85952	19	1239	104991	22
			5	1242	103553	22	1239	97208	21	1239	94249	20	1242	103553	22
	100	100	1	1238	747123	155	1236	786581	162	1234	713437	149	1238	747123	155
			2	1238	769049	160	1238	769147	158	1234	726385	149	1238	769049	160
			3	1236	734372	154	1239	781445	160	1236	749225	153	1236	734372	154
			4	1234	742455	155	1238	781104	160	1238	728220	149	1234	742455	155
			5	1239	771693	159	1234	745833	155	1239	721180	148	1239	771693	159
	100	1.000	1	1236	7513915	1553	1236	7346584	1505	1238	6784554	1384	1236	7513915	1553
			2	1234	6846162	1415	1236	7306146	1501	1236	6861837	1401	1234	6846162	1415
			3	1236	7332730	1508	1236	7552423	1552	1234	6801437	1399	1236	7332730	1508
			4	1236	7321776	1513	1236	7603333	1562	1236	6848565	1402	1236	7321776	1513
			5	1236	7236223	1485	1236	7400664	1520	1236	6811133	1398	1236	7236223	1485
	100	10.000	1	1236	68052742	14152	1236	69381114	14437	1236	64953484	13353	1236	68052742	14152
			2	1236	68244373	14233	1236	67803451	14097	1236	64341697	13386	1236	68244373	14233
			3	1236	67151224	13980	1236	69032556	14384	1236	63948594	13175	1236	67151224	13980
			4	1236	69300690	14475	1236	70454616	14706	1236	64753940	13404	1236	69300690	14475
			5	1236	69830093	14599	1236	70152011	14670	1236	64779672	13377	1236	69830093	14599

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ5 – Modelos de Híbridização.

Tabela 59 – Análise de sensibilidade do *makespan* ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” (variando entre 0 e 30) com os parâmetros definidos na Tab. 25 e número de ciclos = 10, para o problema ABZ5 (resultados obtidos por proc. 2^{}). Solução ótima: 1234 UT.**

Problema	Número de ciclos	% de partículas que sofrem mutação	Rodada	Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ5	10	0	1	1238	6649786	1378	1238	6745250	1396	1238	6405724	1324
			2	1238	6567461	1355	1238	6653496	1359	1239	6139106	1280
			3	1238	6675905	1368	1239	6334695	1305	1239	6248683	1297
			4	1238	6720220	1385	1238	6861799	1409	1238	6395913	1321
			5	1238	6647414	1367	1238	6803934	1415	1238	6637321	1365
	10	10	1	1238	6949991	1439	1238	6743018	1385	1238	6485893	1333
			2	1238	6932935	1424	1238	6528110	1335	1238	6558488	1365
			3	1238	6980527	1446	1236	6747447	1379	1238	6480366	1338
			4	1236	6708296	1402	1238	6833394	1406	1238	6418232	1324
			5	1238	6872471	1425	1236	6500055	1336	1238	6638088	1382
	10	20	1	1236	6777812	1400	1238	7126795	1468	1238	6503898	1345
			2	1236	6946920	1439	1238	7055463	1454	1236	6437365	1324
			3	1236	6742363	1381	1238	6770388	1384	1238	6653158	1372
			4	1238	6860476	1412	1238	6906625	1445	1238	6644034	1385
			5	1238	7013306	1447	1238	6642730	1354	1238	6471328	1335
	10	30	1	1238	7225744	1494	1238	6987176	1442	1238	6772733	1401
			2	1238	6930437	1428	1238	7003879	1433	1238	6840602	1422
			3	1238	6907792	1426	1238	6766504	1398	1238	6695015	1379
			4	1238	6957986	1423	1236	7054184	1466	1239	6816018	1437
			5	1238	6986712	1445	1238	7088585	1456	1238	6876003	1424

Fonte: O autor, 2010.

^{**}Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ5 – Modelos de Híbridização.

Tabela 60 – Análise de sensibilidade do *makespan* ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” (variando entre 40 e 70) com os parâmetros definidos na Tab. 25 e número de ciclos = 10, para o problema ABZ5 (resultados obtidos pelo proc. 2 **). Solução ótima: 1234 UT.

Problema	Número de ciclos	% de partículas que sofrem mutação	Rodada	Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ5	10	40	1	1238	7202864	1495	1236	6999524	1425	1238	7021951	1456
			2	1236	7127296	1472	1236	6987345	1442	1236	6618579	1374
			3	1236	7229963	1494	1238	6882166	1413	1238	6837471	1418
			4	1236	7280389	1501	1238	7029230	1454	1238	6756378	1410
			5	1238	7237842	1486	1238	7188740	1484	1238	7053925	1466
	10	50	1	1236	7371834	1522	1238	7091616	1462	1238	6864976	1440
			2	1236	7473047	1540	1238	6901943	1417	1238	6698716	1393
			3	1236	7659424	1580	1238	6938335	1430	1238	6680644	1379
			4	1236	7246324	1494	1238	7037480	1447	1238	6921511	1431
			5	1238	7618666	1568	1238	6854411	1403	1239	6799794	1408
	10	60	1	1236	7490905	1541	1236	6961133	1421	1238	7053065	1477
			2	1238	7417647	1528	1238	6931873	1432	1236	6797088	1406
			3	1236	7498461	1540	1238	6978500	1430	1236	6689688	1393
			4	1238	7653198	1582	1238	6993487	1438	1238	6871220	1428
			5	1236	7575591	1566	1238	7220000	1490	1238	6816010	1411
	10	70	1	1236	7556121	1567	1236	7411404	1533	1238	6756306	1397
			2	1238	7831153	1611	1238	7111471	1465	1238	7129550	1486
			3	1238	7521515	1557	1238	7148603	1477	1238	7008838	1459
			4	1236	7710189	1599	1238	6838697	1406	1238	6999438	1451
			5	1238	7747079	1596	1238	7137416	1468	1238	6944972	1431

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ5 – Modelos de Híbridização.

Tabela 61 – Análise de sensibilidade do *makespan* ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” (variando entre 80 e 100) com os parâmetros definidos na Tab. 25 e número de ciclos = 10, para o problema ABZ5 (resultados obtidos pelo processador 2^{}). Solução ótima para este problema: 1234 UT.**

Problema	Número de ciclos	% de partículas que sofrem mutação	Rodada	Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ5	10	80	1	1238	8054621	1664	1236	7044178	1441	1236	7070556	1463
			2	1236	7887300	1631	1238	7287391	1506	1238	6920583	1429
			3	1238	7897499	1630	1238	7003756	1441	1238	7257244	1518
			4	1238	7839588	1623	1238	7294916	1512	1238	7286943	1513
			5	1236	7616402	1571	1238	7089454	1449	1236	7129220	1475
	10	90	1	1236	7964036	1643	1236	7460265	1529	1238	7197075	1493
			2	1238	7787577	1600	1238	7263576	1500	1238	7164852	1494
			3	1238	7601605	1570	1238	7360563	1514	1238	7276176	1514
			4	1236	7805346	1615	1236	7259470	1487	1238	7070571	1463
			5	1238	7953381	1635	1238	7295998	1508	1236	7249417	1502
	10	100	1	1236	8259671	1702	1238	7250327	1494	1234	7288346	1506
			2	1238	8329264	1711	1234	7377937	1510	1236	7589839	1582
			3	1238	8600394	1763	1238	7350238	1511	1238	7040303	1455
			4	1238	8138809	1677	1238	7444852	1532	1238	7504042	1564
			5	1236	8300399	1707	1238	7355052	1513	1238	7359929	1535

Fonte: O autor, 2010.

^{**}Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ5 – Modelos de Híbridização.

Tabela 62 – Análise de sensibilidade do *makespan* ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” (variando entre 0 e 30) com os parâmetros definidos na Tab. 26 e número de ciclos = 10, para o problema ABZ5 (resultados obtidos pelo proc. 2 **). Solução ótima: 1234 UT.

Problema	Número de ciclos	% de partículas que sofrem mutação	Rodada	Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ5	10	0	1	1238	6480172	1326	1236	6421668	1314	1236	6692546	1387
			2	1238	6605353	1340	1238	6643273	1358	1239	6578336	1365
			3	1238	6676441	1384	1239	6485131	1337	1238	6716616	1438
			4	1239	6447120	1344	1238	6430220	1319	1238	6747904	1400
			5	1238	6764986	1411	1238	6714823	1370	1238	6577070	1360
	10	10	1	1238	6750589	1390	1238	6849945	1418	1238	7172900	1473
			2	1238	7005564	1461	1238	6710386	1380	1238	7023521	1450
			3	1238	7099046	1452	1236	6513948	1341	1236	6701189	1381
			4	1238	6564897	1347	1238	6707751	1377	1238	7351526	1529
			5	1238	6981747	1439	1238	6595126	1348	1238	6762786	1401
	10	20	1	1238	7231849	1484	1238	6688106	1380	1238	7102341	1474
			2	1238	6737218	1381	1238	6834011	1409	1236	7163713	1474
			3	1238	6947100	1426	1238	6912027	1423	1236	7338504	1515
			4	1238	6914090	1415	1238	6934925	1428	1238	7451451	1551
			5	1238	7135959	1477	1238	7035695	1445	1236	7047990	1448
	10	30	1	1238	7155220	1470	1238	6878725	1410	1238	7236607	1501
			2	1238	7196060	1503	1239	6844253	1408	1238	7062127	1455
			3	1238	6952054	1435	1238	6660074	1387	1236	7190130	1479
			4	1236	7195610	1478	1238	6891022	1410	1236	7031399	1441
			5	1238	7098401	1466	1238	6670278	1371	1238	7291595	1525

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ5 – Modelos de Híbridização.

Tabela 63 – Análise de sensibilidade do *makespan* ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” (variando entre 40 e 70) com os parâmetros definidos na Tab. 26 e número de ciclos = 10, para o problema ABZ5 (resultados obtidos pelo proc. 2 **). Solução ótima: 1234 UT.

Problema	Número de ciclos	% de partículas que sofrem mutação	Rodada	Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ5	10	40	1	1238	7285185	1506	1238	6940731	1440	1236	7471256	1545
			2	1238	7208840	1489	1238	6763130	1383	1238	7041232	1483
			3	1238	7614619	1567	1238	6982271	1436	1236	7203446	1490
			4	1238	7685767	1579	1238	6771749	1397	1238	6973663	1426
			5	1238	7124536	1471	1238	7081791	1476	1236	7712087	1590
	10	50	1	1238	7841546	1621	1238	7002860	1444	1236	7967758	1642
			2	1236	7455251	1543	1238	6817873	1400	1238	7345387	1529
			3	1238	7251043	1507	1236	7100592	1471	1236	8163997	1680
			4	1236	7347784	1520	1238	7076496	1450	1236	7589564	1568
			5	1236	7719861	1586	1236	7044267	1448	1236	7752889	1603
	10	60	1	1236	7847963	1630	1238	7205141	1479	1238	7495841	1544
			2	1238	7834750	1613	1238	7315053	1498	1238	7630465	1577
			3	1238	7835007	1611	1238	7317552	1502	1236	7828896	1612
			4	1236	7990786	1648	1236	6947452	1427	1236	7858605	1626
			5	1236	7653059	1574	1238	6932691	1426	1238	8031057	1668
	10	70	1	1236	7539449	1567	1238	7005084	1445	1236	7964710	1658
			2	1238	7482580	1537	1236	6973818	1428	1236	7727574	1591
			3	1238	7971278	1650	1238	7245972	1501	1238	8331878	1720
			4	1238	7516266	1544	1236	6818440	1403	1238	8124906	1694
			5	1238	7726149	1596	1236	7011368	1462	1236	8014003	1669

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ5 – Modelos de Híbridização.

Tabela 64 – Análise de sensibilidade do *makespan* ao valor do parâmetro “porcentagem de partículas que sofrem mutação por iteração” (variando entre 80 e 100) com os parâmetros definidos na Tab. 26 e número de ciclos = 10, para o problema ABZ5 (resultados obtidos pelo proc. 2 **). Solução ótima: 1234 UT.

Problema	Número de ciclos	% de partículas que sofrem mutação	Rodada	Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ5	10	80	1	1236	8273966	1707	1236	7217891	1491	1236	8333112	1725
			2	1238	7882396	1630	1238	7204021	1487	1238	8263517	1714
			3	1238	7946763	1648	1238	7241597	1496	1236	7878330	1629
			4	1238	7796055	1596	1236	7268026	1493	1238	8384478	1735
			5	1236	8162882	1683	1236	7479630	1551	1236	8033864	1657
	10	90	1	1234	8053940	1660	1238	7305504	1499	1236	8150549	1679
			2	1238	7819135	1609	1238	7259767	1491	1234	8484042	1748
			3	1238	8010827	1649	1238	7121771	1465	1238	8507333	1761
			4	1238	7784339	1612	1238	7216390	1486	1238	7852785	1621
			5	1238	8039071	1645	1238	7027446	1449	1236	8084749	1666
	10	100	1	1238	8404018	1729	1238	7297542	1503	1238	8411719	1733
			2	1236	8323422	1716	1238	7048891	1451	1238	8119559	1675
			3	1238	8136632	1676	1238	7319051	1511	1236	8372441	1732
			4	1236	8240047	1701	1238	7280982	1493	1238	8348937	1720
			5	1236	8495099	1746	1238	7323414	1509	1238	8425773	1743

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ6 – Busca Tabu.

Tabela 65 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema ABZ6 (resultados obtidos pelo processador 2^o). Solução ótima para este problema: 934 UT.

Solução	Valor inicial	Número máximo de iterações sem melhora do <i>makespan</i>				
		100	1.000	10.000	100.000	1.000.000
1	1483	996	948	947	947	947
2	1778	957	957	947	947	947
3	1869	998	948	947	947	947
4	1470	1018	961	947	947	947
5	1646	1013	967	957	957	957
6	1556	951	951	947	947	947
7	1430	971	962	947	947	947
8	1754	971	966	947	947	947
9	1729	997	958	947	947	947
10	1871	979	966	947	947	947
11	1691	1077	966	956	956	956
12	1667	987	947	947	947	947
Melhor solução	1430	951	947	947	947	947
Número total de soluções computadas	XXX	14471	117798	980410	6582835	62602112
Tempo total de CPU	XXX	3	24	202	1356	12883

Fonte: O autor, 2010.

Tabela 66 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema ABZ6 (resultados obtidos pelo processador 2^o). Solução ótima para este problema 943 UT.

Solução	Valor inicial	Tamanho da lista tabu									
		6	7	8	9	10	11	12	13	14	15
1	1483	945	947	947	948	947	948	947	947	947	947
2	1778	947	947	947	948	948	948	945	943	947	948
3	1869	984	957	947	948	947	945	947	947	948	947
4	1470	1018	947	947	945	947	948	947	947	948	947
5	1646	964	947	957	945	947	947	951	948	947	948
6	1556	948	947	947	945	947	947	951	947	948	948
7	1430	1001	947	947	945	945	947	945	951	945	947
8	1754	945	945	947	948	945	945	947	947	945	957
9	1729	996	962	947	947	945	946	947	948	945	947
10	1871	950	957	947	945	952	945	947	948	943	948
11	1691	945	947	956	947	945	946	945	947	947	947
12	1667	967	947	947	945	947	947	958	947	947	945
Melhor solução	1430	945	945	947	945	945	945	945	943	943	945
Nº total de soluções computadas	XXX	628132	745519	980410	970748	894905	999470	847907	946767	1003918	958436
Tempo total de CPU	XXX	130	151	199	197	182	204	175	196	209	200

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ6 – Modelos de Híbridização.

Tabela 67 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 10, para o problema ABZ6 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 943 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ6	10	100	1	947	104145	21	947	80379	17	947	83704	18	948	77793	16
			2	945	94699	20	947	83158	17	948	82260	17	947	74615	16
			3	945	96970	20	947	82843	17	947	78131	16	947	75620	16
			4	945	95677	20	947	88739	18	947	79954	17	947	74406	15
			5	945	102936	21	947	82265	17	947	80749	17	948	76415	16
	10	1.000	1	945	764840	158	947	739178	153	945	778672	161	945	749609	156
			2	945	697299	144	943	786042	162	945	790996	162	945	734679	152
			3	945	735495	152	947	749799	154	945	757048	156	947	742392	154
			4	943	739774	152	943	817048	168	945	824903	170	943	755884	157
			5	945	720886	149	947	757949	156	945	752594	155	947	731743	152
	10	10.000	1	945	6242376	1288	945	7224224	1493	945	6944709	1436	947	6779472	1411
			2	945	5966540	1231	945	6855543	1415	947	6745119	1392	947	6667534	1387
			3	943	6191070	1281	945	6900983	1422	947	7010716	1446	947	6766670	1405
			4	945	5848297	1205	943	7071709	1458	947	6778075	1399	947	6745710	1390
			5	945	5800401	1196	947	6899999	1422	947	6945696	1432	947	6677570	1383
	10	100.000	1	943	50614280	10434	945	62053063	12845	947	62335375	12905	947	62141850	12923
			2	945	45189634	9307	945	62568837	12926	945	62369223	12934	947	62254529	12974
			3	945	41997663	8670	945	62268876	12863	945	62380620	12895	945	62708910	13041
			4	945	50959960	10510	945	62460656	12914	945	62754258	12943	945	62879358	13077
			5	945	50580783	10478	943	62496418	12946	945	62676846	12929	947	62310933	12975

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ6 – Modelos de Híbridização.

Tabela 68 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 100, para o problema ABZ6 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 943 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ6	100	10	1	948	174243	37	947	92283	20	947	80192	18	948	77572	17
			2	958	175105	36	947	92010	20	958	92044	19	958	89689	19
			3	951	172870	37	943	100165	22	948	84648	19	948	82166	18
			4	945	173770	36	952	103022	21	957	79464	18	952	90003	20
			5	948	172093	36	947	93912	20	958	92945	20	952	93562	19
	100	100	1	945	948142	197	943	756036	158	947	715539	148	947	700777	147
			2	945	972616	202	943	747217	154	947	697127	143	947	679650	143
			3	943	954672	198	945	734709	152	947	737008	151	947	694839	144
			4	945	942245	195	947	755979	156	945	678149	142	943	699354	146
			5	945	929711	193	943	742576	154	945	670859	141	947	690888	145
	100	1.000	1	945	7201229	1485	943	7155805	1479	943	6594056	1369	943	6696246	1385
			2	943	6991412	1444	943	7195472	1482	943	6744869	1390	943	6863323	1425
			3	943	7080278	1460	943	7318260	1509	943	6740809	1393	943	6758037	1400
			4	945	7226968	1491	943	7323673	1512	943	6919424	1428	943	6894580	1433
			5	943	7161846	1477	943	7512974	1545	943	6714470	1383	945	6714476	1390
	100	10.000	1	943	58805895	12124	943	69016301	14233	943	64966842	13406	945	64025699	13308
			2	943	59878469	12370	945	67858442	13974	943	65700346	13563	943	65428457	13575
			3	943	57994601	11960	943	68760286	14184	945	65861156	13566	943	64480645	13364
			4	943	58782864	12139	943	69232106	14297	943	65109120	13384	945	64736149	13441
			5	943	58507722	12072	943	67611826	13952	943	64957290	13394	945	63975335	13282

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ6 – Modelos de Híbridização.

Tabela 69 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 10, para o problema ABZ6 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 943 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ6	10	100	1	947	104145	21	945	85680	17	947	90281	19	948	85717	18
			2	945	94699	20	947	82400	17	945	84296	18	945	96340	20
			3	945	96970	20	947	83008	18	947	86746	18	947	91196	19
			4	945	95677	20	947	86248	18	947	85183	18	947	97389	21
			5	945	102936	21	947	87099	18	947	84018	17	945	87374	18
	10	1.000	1	945	764840	158	945	793091	163	945	782106	162	945	792229	165
			2	945	697299	144	945	828469	171	945	787127	163	945	809236	168
			3	945	735495	152	945	765153	158	945	751129	154	943	782393	162
			4	943	739774	152	943	825090	170	945	744460	153	943	799471	166
			5	945	720886	149	945	788475	163	943	798294	165	945	775848	160
	10	10.000	1	945	6242376	1288	945	7110253	1468	945	6904218	1426	945	6820628	1423
			2	945	5966540	1231	945	7201946	1480	945	6921038	1431	945	6955935	1445
			3	943	6191070	1281	945	6903273	1423	945	7069451	1458	945	7157749	1487
			4	945	5848297	1205	943	7088793	1466	945	7015977	1442	945	6929390	1438
			5	945	5800401	1196	947	6781477	1400	947	6898574	1419	945	6893315	1432
	10	100.000	1	943	50614280	10434	947	62376846	12910	943	62458902	12879	943	62524286	12964
			2	945	45189634	9307	943	62687279	12911	945	62620576	12913	945	62121719	12947
			3	945	41997663	8670	945	62597056	12930	945	62389321	12836	945	62857523	13038
			4	945	50959960	10510	945	63092828	12997	945	62556990	12934	943	62625339	13013
			5	945	50580783	10478	945	62214609	12850	945	62648373	12920	945	62319798	12931

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ6 – Modelos de Híbridização.

Tabela 70 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 100, para o problema ABZ6 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 943 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ6	100	10	1	948	174243	37	948	86534	19	966	88738	19	947	94914	21
			2	958	175105	36	948	96533	20	948	97171	21	947	95696	21
			3	951	172870	37	948	92874	21	948	86117	19	948	90216	20
			4	945	173770	36	948	90232	19	948	84829	19	947	96551	21
			5	948	172093	36	947	87954	19	947	94725	20	951	100974	21
	100	100	1	945	948142	197	943	738673	154	945	722726	150	947	771318	161
			2	945	972616	202	947	749604	155	947	732906	150	945	761092	157
			3	943	954672	198	947	751813	155	945	718441	148	945	758995	158
			4	945	942245	195	943	776193	160	947	708133	146	947	746178	154
			5	945	929711	193	943	766626	159	943	703319	145	943	761054	160
	100	1.000	1	945	7201229	1485	943	7378173	1517	943	6812024	1403	943	7163965	1492
			2	943	6991412	1444	943	7346115	1514	945	6675399	1376	943	7149091	1489
			3	943	7080278	1460	943	7280127	1501	945	6908234	1423	943	7328574	1521
			4	945	7226968	1491	943	7280786	1506	943	6883878	1421	943	7357722	1528
			5	943	7161846	1477	943	7314963	1508	943	6965859	1435	943	7106004	1470
	100	10.000	1	943	58805895	12124	945	69263835	14299	943	65630020	13528	945	68748570	14261
			2	943	59878469	12370	943	67762505	14015	943	65836476	13560	945	66760004	13860
			3	943	57994601	11960	943	68612789	14156	943	65577518	13517	943	67445561	13990
			4	943	58782864	12139	943	68493734	14203	943	65524044	13529	943	67566452	14028
			5	943	58507722	12072	943	68167012	14094	943	65553061	13549	943	66553539	13796

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ7 – Busca Tabu.

Tabela 71 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema ABZ7 (resultados obtidos pelo processador 2^o). Solução ótima para este problema: 656 UT.

Solução	Valor inicial	Número máximo de iterações sem melhora do <i>makespan</i>				
		100	1.000	10.000	100.000	1.000.000
1	1464	797	709	673	673	673
2	1186	731	695	673	673	673
3	1260	765	693	683	680	680
4	1461	781	697	672	672	672
5	1411	735	692	664	664	664
6	1325	770	690	678	673	673
7	1394	752	693	681	681	681
8	1283	724	694	681	681	681
9	1276	729	707	680	671	671
10	1314	756	697	679	678	678
11	1305	722	713	693	673	673
12	1382	734	702	674	672	672
Melhor solução	1186	722	690	664	664	664
Número total de soluções computadas	XXX	58332	369944	2336910	11433157	83340148
Tempo total de CPU	XXX	29	188	1180	5826	42812

Fonte: O autor, 2010.

Tabela 72 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema ABZ7 (resultados obtidos pelo processador 2^o). Solução ótima para este problema 656 UT.

Solução	Valor inicial	Tamanho da lista tabu									
		6	7	8	9	10	11	12	13	14	15
1	1464	694	687	673	672	675	674	673	674	678	674
2	1186	729	670	673	673	676	678	670	669	675	676
3	1260	713	682	680	675	673	677	675	673	675	678
4	1461	697	676	672	669	672	670	669	674	673	677
5	1411	680	686	664	672	673	676	675	672	678	673
6	1325	678	674	673	672	668	669	673	676	671	676
7	1394	704	684	681	672	675	675	676	676	672	677
8	1283	723	684	681	671	674	672	676	674	679	676
9	1276	674	683	671	672	670	675	675	674	674	674
10	1314	719	725	678	674	670	672	677	675	676	675
11	1305	732	693	673	670	668	671	673	677	675	675
12	1382	674	675	672	670	674	673	675	673	675	677
Melhor solução	1186	674	670	664	669	668	669	669	669	671	673
Nº total de soluções computadas	XXX	9173243	9388105	11433157	14861536	17277500	14371783	15048259	17682065	14905153	13725782
Tempo total de CPU	XXX	4605	4713	5724	7366	8562	7112	7489	8816	7408	6840

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ7 – Modelos de Híbridização.

Tabela 73 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 10, para o problema ABZ7 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 656 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ7	10	100	1	701	317688	162	691	234972	120	704	234785	119	710	198156	101
			2	693	315317	162	694	234147	119	693	218361	112	711	190250	98
			3	692	300104	152	695	215598	110	702	213882	109	706	198407	102
			4	692	309625	157	697	235109	120	699	204401	104	698	206248	106
			5	693	291324	147	696	260081	131	696	245972	125	706	209839	108
	10	1.000	1	674	1965086	998	672	1682729	853	677	1649536	838	678	1534538	784
			2	670	2058309	1038	671	1848979	941	679	1822673	923	671	1432418	732
			3	672	1776233	899	676	1711351	869	674	1755653	885	680	1528722	776
			4	675	1771407	898	668	1746496	885	682	1564662	789	678	1471504	751
			5	679	1822085	922	672	1683532	853	674	1766952	895	677	1524454	777
	10	10.000	1	668	13098891	6607	664	13131717	6656	664	12957582	6558	664	12169315	6195
			2	668	12596867	6355	664	12123569	6148	664	11882406	6000	664	10854844	5525
			3	667	14147757	7134	664	12586727	6388	664	12739373	6448	664	11530304	5868
			4	667	12510263	6310	664	12577989	6365	663	13621960	6843	664	11789190	5985
			5	667	12418872	6255	664	13501614	6830	664	12929440	6538	664	11480802	5847
	10	100.000	1	665	82309991	41728	664	99592669	50595	664	97036371	49359	664	97161387	49599
			2	666	75740919	38284	664	97454995	49684	664	97522118	49329	664	94289328	48196
			3	665	84624384	42862	664	98887349	50420	664	98834718	50613	664	94520593	48393
			4	665	77593847	39406	664	100143444	50742	664	96778883	49353	664	96100747	48972
			5	665	83421308	42302	664	101917697	51897	664	96644230	49207	664	92241045	47537

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ7 – Modelos de Híbridização.

Tabela 74 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 100, para o problema ABZ7 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 656 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ7	100	10	1	729	766072	391	700	254832	131	715	199746	106	692	182908	98
			2	738	762206	387	694	244444	126	702	195971	105	725	176023	94
			3	728	762479	388	716	255241	133	697	221947	117	700	193360	101
			4	729	764294	387	706	267768	137	733	211472	112	697	192252	101
			5	716	763481	387	701	254968	131	730	235983	121	708	196382	103
	100	100	1	687	2991865	1516	679	1622873	822	689	1368707	698	677	1350172	679
			2	693	3019652	1531	673	1594401	807	682	1320886	679	687	1424536	715
			3	691	2953902	1498	681	1576547	789	679	1351266	695	680	1280302	653
			4	689	2978534	1506	684	1613305	805	690	1309380	680	692	1290060	654
			5	693	3086143	1565	679	1598263	805	690	1323327	685	677	1245350	642
	100	1.000	1	674	17970021	9084	668	12767136	6447	670	11219098	5732	665	10987580	5553
			2	668	18490518	9359	666	12802076	6403	674	11283539	5799	667	11337093	5696
			3	669	18733052	9457	665	13077719	6552	664	10852477	5556	671	10118811	5188
			4	671	18567432	9393	668	13037315	6486	669	12204390	6120	673	11217209	5659
			5	672	18140471	9181	670	12198327	6152	673	11033776	5646	669	10917527	5487
	100	10.000	1	666	124536162	62844	664	112905733	56275	664	98120431	50306	664	100032917	50504
			2	665	124013417	62685	664	107651375	53939	662	101348717	51538	664	98286368	49887
			3	666	120675736	60848	664	106766040	53550	664	104858063	53088	663	100808385	51056
			4	665	121140529	61084	662	111008388	55465	663	107509101	54384	664	101098600	50896
			5	660	122603360	61866	664	104855252	52618	663	104053302	52929	661	97655666	49366

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ7 – Modelos de Híbridização.

Tabela 75 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 10, para o problema ABZ7 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 656 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ7	10	100	1	701	317688	162	694	213076	107	698	227681	117	705	219826	114
			2	693	315317	162	683	229530	116	700	222666	115	691	199812	103
			3	692	300104	152	694	223597	114	689	221800	114	692	249965	127
			4	692	309625	157	691	198673	101	700	243395	125	685	259208	132
			5	693	291324	147	691	224194	113	704	213757	109	677	241374	123
	10	1.000	1	674	1965086	998	671	1571393	792	675	1721177	881	674	1591294	808
			2	670	2058309	1038	676	1692489	847	673	1696410	861	672	1599005	812
			3	672	1776233	899	674	1562700	780	676	1794750	907	675	1680170	854
			4	675	1771407	898	672	1663764	830	675	1784510	909	682	1548377	787
			5	679	1822085	922	669	1656242	833	671	1659493	845	672	1577229	801
	10	10.000	1	668	13098891	6607	667	12035648	6053	664	13096772	6652	664	14019856	7074
			2	668	12596867	6355	664	13008018	6500	664	12206929	6205	666	13346321	6731
			3	667	14147757	7134	666	13318840	6645	664	12292202	6255	668	12885996	6530
			4	667	12510263	6310	667	13117779	6576	664	11684354	5922	666	13510445	6802
			5	667	12418872	6255	666	12731914	6375	664	11919668	6071	668	12909878	6550
	10	100.000	1	665	82309991	41728	665	102765888	51595	664	99665582	50832	667	100798352	51098
			2	666	75740919	38284	664	99916498	50024	664	99804857	51253	665	99775337	50741
			3	665	84624384	42862	665	97758195	49316	664	96049085	49010	665	104529833	52781
			4	665	77593847	39406	665	99726576	50181	664	100248517	51298	658	99276632	50362
			5	665	83421308	42302	662	99660314	50167	664	104865828	53621	664	98312521	49653

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ7 – Modelos de Híbridização.

Tabela 76 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 100, para o problema ABZ7 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 656 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ7	100	10	1	729	766072	391	729	766072	391	729	766072	391	729	766072	391
			2	738	762206	387	738	762206	387	738	762206	387	738	762206	387
			3	728	762479	388	728	762479	388	728	762479	388	728	762479	388
			4	729	764294	387	729	764294	387	729	764294	387	729	764294	387
			5	716	763481	387	716	763481	387	716	763481	387	716	763481	387
	100	100	1	687	2991865	1516	687	2991865	1516	687	2991865	1516	687	2991865	1516
			2	693	3019652	1531	693	3019652	1531	693	3019652	1531	693	3019652	1531
			3	691	2953902	1498	691	2953902	1498	691	2953902	1498	691	2953902	1498
			4	689	2978534	1506	689	2978534	1506	689	2978534	1506	689	2978534	1506
			5	693	3086143	1565	693	3086143	1565	693	3086143	1565	693	3086143	1565
	100	1.000	1	674	17970021	9084	674	17970021	9084	674	17970021	9084	674	17970021	9084
			2	668	18490518	9359	668	18490518	9359	668	18490518	9359	668	18490518	9359
			3	669	18733052	9457	669	18733052	9457	669	18733052	9457	669	18733052	9457
			4	671	18567432	9393	671	18567432	9393	671	18567432	9393	671	18567432	9393
			5	672	18140471	9181	672	18140471	9181	672	18140471	9181	672	18140471	9181
	100	10.000	1	666	124536162	62844	666	124536162	62844	666	124536162	62844	666	124536162	62844
			2	665	124013417	62685	665	124013417	62685	665	124013417	62685	665	124013417	62685
			3	666	120675736	60848	666	120675736	60848	666	120675736	60848	666	120675736	60848
			4	665	121140529	61084	665	121140529	61084	665	121140529	61084	665	121140529	61084
			5	660	122603360	61866	660	122603360	61866	660	122603360	61866	660	122603360	61866

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ8 – Busca Tabu.

Tabela 77 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema ABZ8 (resultados obtidos pelo processador 2^o). Solução ótima para este problema: 645 UT.

Solução	Valor inicial	Número máximo de iterações sem melhora do <i>makespan</i>				
		100	1.000	10.000	100.000	1.000.000
1	1382	746	712	696	696	693
2	1334	742	694	694	685	685
3	1283	798	719	693	693	693
4	1534	779	693	685	685	685
5	1253	751	704	693	680	680
6	1522	720	710	676	675	675
7	1308	820	777	703	703	703
8	1420	780	732	692	683	683
9	1524	809	711	681	681	681
10	1280	759	712	689	687	687
11	1467	755	721	693	693	693
12	1402	771	715	688	681	681
Melhor solução	1253	720	693	676	675	675
Número total de soluções computadas	XXX	49229	383297	2466778	14104400	94655547
Tempo total de CPU	XXX	26	198	1258	7211	48669

Fonte: O autor, 2010.

Tabela 78 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema ABZ8 (resultados obtidos pelo processador 2^o). Solução ótima para este problema 645 UT.

Solução	Valor inicial	Tamanho da lista tabu									
		6	7	8	9	10	11	12	13	14	15
1	1382	722	720	696	682	688	685	684	687	684	690
2	1334	692	691	685	682	686	683	684	682	686	692
3	1283	697	719	693	697	682	684	683	689	677	685
4	1534	693	698	685	695	684	684	691	691	690	684
5	1253	714	693	680	691	680	687	688	694	692	683
6	1522	696	692	675	681	682	682	687	686	689	687
7	1308	760	701	703	683	686	688	683	683	686	686
8	1420	701	697	683	682	689	687	689	690	682	689
9	1524	703	683	681	710	682	684	681	691	682	682
10	1280	695	686	687	687	687	693	690	687	690	692
11	1467	704	682	693	683	680	688	686	692	686	686
12	1402	711	698	681	682	688	686	685	686	682	692
Melhor solução	1253	692	682	675	681	680	682	681	682	677	682
Nº total de soluções computadas	XXX	10060503	10158215	14104400	14468326	14543705	18456839	18595992	15643196	18320165	16635182
Tempo total de CPU	XXX	5003	5085	7019	7162	7211	9145	7737	7737	9122	8237

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ8 – Modelos de Híbridização.

Tabela 79 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 10, para o problema ABZ8 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 645 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ8	10	100	1	713	320170	164	713	320170	164	713	320170	164	713	320170	164
			2	710	332824	172	710	332824	172	710	332824	172	710	332824	172
			3	707	304026	156	707	304026	156	707	304026	156	707	304026	156
			4	711	296060	152	711	296060	152	711	296060	152	711	296060	152
			5	713	310582	159	713	310582	159	713	310582	159	713	310582	159
	10	1.000	1	689	2036595	1031	689	2036595	1031	689	2036595	1031	689	2036595	1031
			2	683	2183506	1105	683	2183506	1105	683	2183506	1105	683	2183506	1105
			3	687	2116671	1070	687	2116671	1070	687	2116671	1070	687	2116671	1070
			4	687	2026437	1026	687	2026437	1026	687	2026437	1026	687	2026437	1026
			5	690	1889161	957	690	1889161	957	690	1889161	957	690	1889161	957
	10	10.000	1	672	12693064	6396	672	12693064	6396	672	12693064	6396	672	12693064	6396
			2	674	12591536	6339	674	12591536	6339	674	12591536	6339	674	12591536	6339
			3	674	13209931	6663	674	13209931	6663	674	13209931	6663	674	13209931	6663
			4	679	11998512	6053	679	11998512	6053	679	11998512	6053	679	11998512	6053
			5	674	12982182	6546	674	12982182	6546	674	12982182	6546	674	12982182	6546
	10	100.000	1	674	89392018	45365	674	89392018	45365	674	89392018	45365	674	89392018	45365
			2	672	96196327	48683	672	96196327	48683	672	96196327	48683	672	96196327	48683
			3	678	94530714	47942	678	94530714	47942	678	94530714	47942	678	94530714	47942
			4	673	92230113	46670	673	92230113	46670	673	92230113	46670	673	92230113	46670
			5	676	92308775	46777	676	92308775	46777	676	92308775	46777	676	92308775	46777

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ8 – Modelos de Híbridização.

Tabela 80 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 100, para o problema ABZ8 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 645 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ8	100	10	1	748	756842	394	748	756842	394	748	756842	394	748	756842	394
			2	745	764871	394	745	764871	394	745	764871	394	745	764871	394
			3	751	751787	389	751	751787	389	751	751787	389	751	751787	389
			4	742	747789	387	742	747789	387	742	747789	387	742	747789	387
			5	731	748256	386	731	748256	386	731	748256	386	731	748256	386
	100	100	1	707	3069905	1575	707	3069905	1575	707	3069905	1575	707	3069905	1575
			2	706	3118658	1600	706	3118658	1600	706	3118658	1600	706	3118658	1600
			3	703	3029116	1555	703	3029116	1555	703	3029116	1555	703	3029116	1555
			4	709	3077665	1580	709	3077665	1580	709	3077665	1580	709	3077665	1580
			5	702	2935624	1508	702	2935624	1508	702	2935624	1508	702	2935624	1508
	100	1.000	1	678	19481521	9888	678	19481521	9888	678	19481521	9888	678	19481521	9888
			2	681	19698382	9989	681	19698382	9989	681	19698382	9989	681	19698382	9989
			3	682	19476240	9890	682	19476240	9890	682	19476240	9890	682	19476240	9890
			4	682	19120322	9714	682	19120322	9714	682	19120322	9714	682	19120322	9714
			5	681	19619129	9960	681	19619129	9960	681	19619129	9960	681	19619129	9960
	100	10.000	1	673	129531270	65298	673	129531270	65298	673	129531270	65298	673	129531270	65298
			2	671	130526900	65856	671	130526900	65856	671	130526900	65856	671	130526900	65856
			3	674	134808840	67916	674	134808840	67916	674	134808840	67916	674	134808840	67916
			4	671	133293379	67147	671	133293379	67147	671	133293379	67147	671	133293379	67147
			5	673	130840717	66016	673	130840717	66016	673	130840717	66016	673	130840717	66016

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ8 – Modelos de Híbridização.

Tabela 81 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 10, para o problema ABZ8 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 645 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ8	10	100	1	713	320170	164	713	320170	164	713	320170	164	713	320170	164
			2	710	332824	172	710	332824	172	710	332824	172	710	332824	172
			3	707	304026	156	707	304026	156	707	304026	156	707	304026	156
			4	711	296060	152	711	296060	152	711	296060	152	711	296060	152
			5	713	310582	159	713	310582	159	713	310582	159	713	310582	159
	10	1.000	1	689	2036595	1031	689	2036595	1031	689	2036595	1031	689	2036595	1031
			2	683	2183506	1105	683	2183506	1105	683	2183506	1105	683	2183506	1105
			3	687	2116671	1070	687	2116671	1070	687	2116671	1070	687	2116671	1070
			4	687	2026437	1026	687	2026437	1026	687	2026437	1026	687	2026437	1026
			5	690	1889161	957	690	1889161	957	690	1889161	957	690	1889161	957
	10	10.000	1	672	12693064	6396	672	12693064	6396	672	12693064	6396	672	12693064	6396
			2	674	12591536	6339	674	12591536	6339	674	12591536	6339	674	12591536	6339
			3	674	13209931	6663	674	13209931	6663	674	13209931	6663	674	13209931	6663
			4	679	11998512	6053	679	11998512	6053	679	11998512	6053	679	11998512	6053
			5	674	12982182	6546	674	12982182	6546	674	12982182	6546	674	12982182	6546
	10	100.000	1	674	89392018	45365	674	89392018	45365	674	89392018	45365	674	89392018	45365
			2	672	96196327	48683	672	96196327	48683	672	96196327	48683	672	96196327	48683
			3	678	94530714	47942	678	94530714	47942	678	94530714	47942	678	94530714	47942
			4	673	92230113	46670	673	92230113	46670	673	92230113	46670	673	92230113	46670
			5	676	92308775	46777	676	92308775	46777	676	92308775	46777	676	92308775	46777

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ8 – Modelos de Híbridização.

Tabela 82 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 100, para o problema ABZ8 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 645 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ8	100	10	1	748	756842	394	748	756842	394	748	756842	394	748	756842	394
			2	745	764871	394	745	764871	394	745	764871	394	745	764871	394
			3	751	751787	389	751	751787	389	751	751787	389	751	751787	389
			4	742	747789	387	742	747789	387	742	747789	387	742	747789	387
			5	731	748256	386	731	748256	386	731	748256	386	731	748256	386
	100	100	1	707	3069905	1575	707	3069905	1575	707	3069905	1575	707	3069905	1575
			2	706	3118658	1600	706	3118658	1600	706	3118658	1600	706	3118658	1600
			3	703	3029116	1555	703	3029116	1555	703	3029116	1555	703	3029116	1555
			4	709	3077665	1580	709	3077665	1580	709	3077665	1580	709	3077665	1580
			5	702	2935624	1508	702	2935624	1508	702	2935624	1508	702	2935624	1508
	100	1.000	1	678	19481521	9888	678	19481521	9888	678	19481521	9888	678	19481521	9888
			2	681	19698382	9989	681	19698382	9989	681	19698382	9989	681	19698382	9989
			3	682	19476240	9890	682	19476240	9890	682	19476240	9890	682	19476240	9890
			4	682	19120322	9714	682	19120322	9714	682	19120322	9714	682	19120322	9714
			5	681	19619129	9960	681	19619129	9960	681	19619129	9960	681	19619129	9960
	100	10.000	1	673	129531270	65298	673	129531270	65298	673	129531270	65298	673	129531270	65298
			2	671	130526900	65856	671	130526900	65856	671	130526900	65856	671	130526900	65856
			3	674	134808840	67916	674	134808840	67916	674	134808840	67916	674	134808840	67916
			4	671	133293379	67147	671	133293379	67147	671	133293379	67147	671	133293379	67147
			5	673	130840717	66016	673	130840717	66016	673	130840717	66016	673	130840717	66016

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ9 – Busca Tabu.

Tabela 83 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema ABZ9 (resultados obtidos pelo processador 2^o). Solução ótima para este problema: 661 UT.

Solução	Valor inicial	Número máximo de iterações sem melhora do <i>makespan</i>				
		100	1.000	10.000	100.000	1.000.000
1	1497	798	727	703	701	701
2	1628	810	742	708	700	700
3	1522	758	715	698	691	691
4	1430	863	753	699	692	692
5	1245	822	719	719	696	696
6	1323	772	752	707	700	700
7	1205	783	736	722	715	715
8	1381	860	738	705	705	705
9	1527	840	731	724	724	724
10	1310	779	729	714	704	704
11	1379	778	734	734	706	706
12	1451	755	722	703	703	703
Melhor solução	1205	755	715	698	691	691
Número total de soluções computadas	XXX	59387	389615	2299998	15316376	101624458
Tempo total de CPU	XXX	31	198	1150	7613	50459

Fonte: O autor, 2010.

Tabela 84 – Análise de sensibilidade do *makespan* ao valor do parâmetro “tamanho da lista tabu” do método Busca Tabu, com os parâmetros fixos definidos na Tab. 24, para o problema ABZ9 (resultados obtidos pelo processador 2^o). Solução ótima para este problema 661 UT.

Solução	Valor inicial	Tamanho da lista tabu									
		6	7	8	9	10	11	12	13	14	15
1	1497	759	795	701	691	692	692	696	699	700	697
2	1628	766	696	700	701	697	698	692	698	691	695
3	1522	719	722	691	691	702	696	695	693	688	697
4	1430	782	719	692	695	699	689	698	695	696	698
5	1245	743	700	696	823	694	695	691	691	700	699
6	1323	741	744	700	697	691	700	693	691	700	693
7	1205	730	709	715	703	692	698	697	691	696	694
8	1381	761	705	705	696	691	692	691	696	698	706
9	1527	701	754	724	704	694	699	697	732	691	691
10	1310	710	776	704	691	692	697	696	695	702	696
11	1379	720	797	706	692	698	700	698	693	700	697
12	1451	743	711	703	689	691	691	696	695	700	697
Melhor solução	1205	701	696	691	689	691	689	691	691	688	691
Nº total de soluções computadas	XXX	9800713	9841819	15316376	15416949	19492841	17771649	21786964	18344690	19409433	21015264
Tempo total de CPU	XXX	4892	4897	7528	7584	9513	8682	10639	9058	9514	10269

Fonte: O autor, 2010.

^{**}Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ9 – Modelos de Híbridização.

Tabela 85 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 10, para o problema ABZ9 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 661 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ9	10	100	1	736	295902	151	727	248645	125	732	247269	126	726	209533	106
			2	727	319736	164	720	253385	127	736	280231	143	733	230257	117
			3	707	299878	154	718	244312	122	721	233963	120	723	211860	108
			4	732	322081	166	721	254211	127	747	249686	128	716	226729	114
			5	716	312649	159	722	258825	130	731	267146	136	733	239576	121
	10	1.000	1	710	1912971	962	690	1833725	896	693	1826363	913	694	1744810	865
			2	698	2037961	1027	690	1936916	954	690	1957390	979	696	1635612	803
			3	702	1987713	999	689	1846155	907	694	1885968	940	691	1643019	813
			4	700	2105056	1061	688	1923259	943	699	1824775	912	700	1652575	815
			5	699	1997638	1009	691	2000806	981	701	2100938	1057	694	1701210	838
	10	10.000	1	688	14854362	7415	688	15380034	7488	690	14465838	7175	689	13579643	6655
			2	689	14896239	7409	688	13654257	6639	688	13404916	6644	685	12957569	6353
			3	690	12980708	6490	688	14183303	6944	688	14091771	6991	693	12788043	6295
			4	689	15027584	7504	688	14771628	7244	689	14198278	7061	691	13446935	6578
			5	688	14296065	7137	688	14607007	7109	690	14763904	7333	687	14282667	6987
	10	100.000	1	687	98348581	49114	688	113946229	55702	688	112686898	56243	687	107951304	52722
			2	687	87256994	43999	688	112851348	55215	686	110736563	54933	688	107881871	52890
			3	686	93091722	46456	685	109109473	53586	686	113845742	56473	690	109548170	53672
			4	688	91301143	45590	688	109307379	53675	686	111099475	55298	686	105107436	51666
			5	686	90567095	45026	687	112604275	55064	686	108026222	53825	687	105071935	51519

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ9 – Modelos de Híbridização.

Tabela 86 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 25 e 27, e número de ciclos = 100, para o problema ABZ9 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 661 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ9	100	10	1	762	695886	364	762	695886	364	762	695886	364	762	695886	364
			2	779	717307	374	779	717307	374	779	717307	374	779	717307	374
			3	774	702984	366	774	702984	366	774	702984	366	774	702984	366
			4	772	702132	369	772	702132	369	772	702132	369	772	702132	369
			5	776	712540	371	776	712540	371	776	712540	371	776	712540	371
	100	100	1	722	3130864	1609	722	3130864	1609	722	3130864	1609	722	3130864	1609
			2	723	3110440	1602	723	3110440	1602	723	3110440	1602	723	3110440	1602
			3	729	3118961	1601	729	3118961	1601	729	3118961	1601	729	3118961	1601
			4	719	3073797	1581	719	3073797	1581	719	3073797	1581	719	3073797	1581
			5	721	3109854	1596	721	3109854	1596	721	3109854	1596	721	3109854	1596
	100	1.000	1	692	19978970	10096	692	19978970	10096	692	19978970	10096	692	19978970	10096
			2	692	20612505	10416	692	20612505	10416	692	20612505	10416	692	20612505	10416
			3	691	19794150	9996	691	19794150	9996	691	19794150	9996	691	19794150	9996
			4	695	19768071	10001	695	19768071	10001	695	19768071	10001	695	19768071	10001
			5	694	20451859	10349	694	20451859	10349	694	20451859	10349	694	20451859	10349
	100	10.000	1	688	143587690	71734	688	143587690	71734	688	143587690	71734	688	143587690	71734
			2	684	139538364	69764	684	139538364	69764	684	139538364	69764	684	139538364	69764
			3	686	138507721	69088	686	138507721	69088	686	138507721	69088	686	138507721	69088
			4	686	137966527	68821	686	137966527	68821	686	137966527	68821	686	137966527	68821
			5	687	136745562	68192	687	136745562	68192	687	136745562	68192	687	136745562	68192

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ9 – Modelos de Híbridização.

Tabela 87 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 10, para o problema ABZ9 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 661 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Colisão de Partículas			Aplicação híbrida sucessiva			Vizinhança híbrida			Vizinhança híbrida melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ9	10	100	1	736	295902	151	725	261839	131	728	241018	124	744	252883	127
			2	727	319736	164	715	263933	132	734	229571	117	719	264062	132
			3	707	299878	154	726	244327	122	729	256201	131	728	273886	137
			4	732	322081	166	707	258624	130	735	266938	136	728	300296	150
			5	716	312649	159	723	282952	141	728	251493	128	740	257478	129
	10	1.000	1	710	1912971	962	693	1803202	882	693	1941932	969	694	1967630	968
			2	698	2037961	1027	694	1897627	929	697	1847124	925	690	1933460	948
			3	702	1987713	999	691	1891665	933	692	1980313	986	698	1855159	914
			4	700	2105056	1061	694	1879692	922	698	1995530	999	697	2079717	1026
			5	699	1997638	1009	702	2178800	1074	695	1685154	844	691	2064363	1021
	10	10.000	1	688	14854362	7415	687	15471982	7515	691	14439775	7170	688	15475484	7565
			2	689	14896239	7409	685	13421986	6506	688	13594831	6784	688	13842785	6753
			3	690	12980708	6490	688	14719826	7160	688	14530600	7213	688	15173889	7399
			4	689	15027584	7504	687	14712960	7171	690	14297685	7110	688	14066305	6879
			5	688	14296065	7137	689	14086886	6847	688	14185584	7046	688	15269873	7489
	10	100.000	1	687	98348581	49114	685	111024761	54030	688	112112394	55510	686	112161300	54789
			2	687	87256994	43999	682	109652177	53844	686	113466383	56216	687	108575062	52987
			3	686	93091722	46456	685	116457631	56707	686	115447059	57194	682	113157343	55546
			4	688	91301143	45590	688	109674696	53408	687	110154261	54834	690	110861524	54447
			5	686	90567095	45026	686	109947757	53895	683	114105820	56604	685	110808745	54296

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB

Problema: ABZ9 – Modelos de Híbridização.

Tabela 88 – Análise de sensibilidade do *makespan* ao valor do parâmetro “número máximo de iterações sem melhora do *makespan*” com os parâmetros definidos nas Tabs. 26 e 27, e número de ciclos = 100, para o problema ABZ9 (resultados obtidos pelo processador 2 **). Solução ótima para este problema: 661 UT.

Problema	Número de ciclos	Número máximo de iterações sem melhora	Rodada	Multi Colisão de Partículas Similar			Aplicação Híbrida Sucessiva			Vizinhança Híbrida			Vizinhança Híbrida Melhorada		
				Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)	Makespan	Número de soluções computadas	Tempo de CPU (s)
ABZ9	100	10	1	762	695886	364	762	695886	364	762	695886	364	762	695886	364
			2	779	717307	374	779	717307	374	779	717307	374	779	717307	374
			3	774	702984	366	774	702984	366	774	702984	366	774	702984	366
			4	772	702132	369	772	702132	369	772	702132	369	772	702132	369
			5	776	712540	371	776	712540	371	776	712540	371	776	712540	371
	100	100	1	722	3130864	1609	722	3130864	1609	722	3130864	1609	722	3130864	1609
			2	723	3110440	1602	723	3110440	1602	723	3110440	1602	723	3110440	1602
			3	729	3118961	1601	729	3118961	1601	729	3118961	1601	729	3118961	1601
			4	719	3073797	1581	719	3073797	1581	719	3073797	1581	719	3073797	1581
			5	721	3109854	1596	721	3109854	1596	721	3109854	1596	721	3109854	1596
	100	1.000	1	692	19978970	10096	692	19978970	10096	692	19978970	10096	692	19978970	10096
			2	692	20612505	10416	692	20612505	10416	692	20612505	10416	692	20612505	10416
			3	691	19794150	9996	691	19794150	9996	691	19794150	9996	691	19794150	9996
			4	695	19768071	10001	695	19768071	10001	695	19768071	10001	695	19768071	10001
			5	694	20451859	10349	694	20451859	10349	694	20451859	10349	694	20451859	10349
	100	10.000	1	688	143587690	71734	688	143587690	71734	688	143587690	71734	688	143587690	71734
			2	684	139538364	69764	684	139538364	69764	684	139538364	69764	684	139538364	69764
			3	686	138507721	69088	686	138507721	69088	686	138507721	69088	686	138507721	69088
			4	686	137966527	68821	686	137966527	68821	686	137966527	68821	686	137966527	68821
			5	687	136745562	68192	687	136745562	68192	687	136745562	68192	687	136745562	68192

Fonte: O autor, 2010.

**Processador 2: Intel Xeon CPU E5410 2.33GHz 4,00GB