UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
MESTRADO

CELSO DE SOUZA TCHAO

HEURISTICAS PARA O PROBLEMA DE ESCALONAMENTO DE PROJETOS COM RESTRIÇÃO DE RECURSOS

Niterói
2007

CELSO DE SOUZA TCHAO

HEURISTICAS PARA O PROBLEMA DE ESCALONAMENTO DE PROJETOS COM RESTRIÇÃO DE RECURSOS

Dissertação apresentada no Curso de Mestrado em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Otimização Combinatória e Inteligência Artificial.

Orientadora: Profª Simone L. Martins, D. Sc.

Niterói

2007

T249 Tchao, Celso de Souza.
Heurística para o problema de escalonamento de projetos com restrição de recursos / Celso de Souza Tchao. – Niterói,. RJ : [s.n.], 2007.
65 f.

Orientador: Simone L. Martins.
Dissertação (Mestrado em Computação) - Universidade Federal Fluminense, 2007.
1. Escalonamento de tarefas. 2. Heurística - Escalonamento. 3. Gerência de projetos. 4. Metaheurística. I. Título.

CDD 005.136

CELSO DE SOUZA TCHAO

HEURISTICAS PARA O PROBLEMA DE ESCALONAMENTO DE PROJETOS COM RESTRIÇÃO DE RECURSOS

Dissertação apresentada no Curso de Mestrado em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Otimização Combinatória e Inteligência Artificial.

Aprovada em 30 de janeiro de 2007.

BANCA EXAMINADORA

_____

Profª Simone L. Martins, D. Sc. – IC/UFF (Presidente)

_____

Prof Luiz Satoru Ochi, D. Sc. – IC/UFF

_____

Profª Adriana C. F. Alvim, D.Sc - DIA/UNIRIO

Niterói

2 0 0 7

A Deus, por tudo.

Aos familiares, pelo permanente apoio, e pelas referências de amor e fé.

À professora Simone Martins, muito é pouco: pela admirável competência e conhecimento que imprimiu na orientação, e pela firmeza e dedicação na condução deste projeto.

Aos professores Celso Ribeiro e Luiz Satoru, agradecimentos pela inspiração e incentivo.

RESUMO

Este trabalho aborda o Problema de Escalonamento de Projetos com Restrição de Recursos - RCPSP (Resource-Constrained Project Scheduling Problem), buscando uma abordagem heurística para um dos processos da Gerência de Projetos, o de Gerenciamento do Prazo do Projeto, cuja etapa de Desenvolvimento do Cronograma contempla a montagem do cronograma, a estimativa de tempo e de custo, além do uso adequado de recursos no tempo considerado. A versão clássica do problema é a de tornar mínimo o tempo total (makespan) de execução de um projeto, levando-se em consideração as restrições de precedência entre as atividades e a disponibilidade dos recursos - escassos e limitados - requeridos por cada uma. Este trabalho incluiu duas variações existentes sobre o RCPSP tradicional: a multiplicidade de tipos de recursos e a multiplicidade de modos possíveis para a execução de cada atividade, que se distinguem mutuamente pelo tempo de execução e pela demanda de recursos. Neste trabalho são descritos comportamentos de heurísticas e metaheurísticas aplicadas na procura de soluções correspondentes a valores ótimos ou próximos de ótimos. Entre as heurísticas, encontram-se os modelos Pert, CPM, SGS Serial e Paralelo e X-Pass (Single e Multi). As metaheurísticas descritas foram selecionadas entre as mais conhecidas, utilizadas em pesquisas recentes. Propõe-se uma solução para o problema, utilizando a metaheurística Busca Tabu com Reconexão de Caminhos. Os resultados computacionais são mostrados e comparados com resultados obtidos por outras abordagens do RCPSP.

Palavras-chave: Heurística, Gerência de Projetos, Cronograma.

ABSTRACT

This work considers the Resource-Constrained Project Scheduling Problem – RCPSP**,** intending a heuristic approach about this Project Management process, the Time and Cost Management, which Project Schedule Development comprehends the timetable construction, the time and cost estimation, and the appropriate resources use over the time range considered. The problem target, in the classic version, is to minimize the project makespan, considering the precedence among activities and the limited resources availability required by each activity. This work includes two features about the traditional RCPSP: multiplicity of kind resources and multiplicity of modes for the execution of each activity, which are distinguished from each other by the duration and the resources demand. In this approach, we observed the behavior of heuristics and metaheuristics applied in the search of solutions corresponding to optimal values or close to them. Pert, CPM, SGS (Serial and Parallel) and X-Pass (Single and Multi) were chosen among the heuristics to be explained. The metaheuristics described were selected from the most known set used in recent researches. This work proposes a solution using Tabu Search and Path Relinking. The obtained results are shown and compared with other results reached by different approaches of the RCPSP.

Keywords: Heuristic, Project Management, Schedule, Timetable.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

O Problema de Escalonamento de Tarefas de Projetos com Restrições de Recursos (RCPSP – Resource Constrained Project Scheduling Problem) refere-se à obtenção do menor makespan, ou o menor tempo de conclusão, de um conjunto de tarefas pertencentes a um determinado projeto, onde cada tarefa possui uma duração estabelecida, e onde devem ser respeitadas as restrições de precedência (uma atividade só é iniciada após a execução das antecessoras) e as de recursos – limitados – que cada tarefa necessita para sua realização. Outros fatores podem gerar variações do problema [27] e [34]: alternativamente ao objetivo de minimizar o makespan, pode-se buscar a minimização do custo ou a maximização da qualidade; os recursos podem ser subdivididos em tipos distintos, além de poderem ser renováveis ou não-renováveis; as regras de precedência entre duas atividades podem se relacionar pelos instantes de início-fim / inicio-inicio / fim-início / fim-fim de cada uma delas; podem existir intervalos de tempo (máximos e mínimos) entre atividades com relação de precedência; podem ocorrer preempções na execução de uma determinada atividade, etc. Este problema pertence à classe NP-Difícil [2] e, por conseguinte, torna interessante a utilização de métodos heurísticos na obtenção de soluções ótimas ou próximas de ótimas. Uma das variações do RCPSP, que é discutida neste trabalho, é o modelo multi-modo, ou seja, a partir de um conjunto de atividades e de um conjunto de recursos disponíveis, são estabelecidas diferentes configurações para a execução de uma mesma atividade. Inicialmente, para cada uma das atividades, é estabelecido um número determinado de configurações (ou modos) em que ela pode ser executada. Cada um destes modos obedece a dois parâmetros para a execução da tarefa: tempo de duração da atividade e quantidade de recursos necessários e compatíveis com a duração correspondente. Além disso, os recursos são divididos em dois tipos, que dependem da sua forma de consumo no decorrer do projeto: recursos renováveis, que não se esgotam com a utilização (ex.: mão-de-obra, equipamento), e recursos não renováveis, que se esgotam com a utilização (ex.: orçamento, matéria prima). Portanto, como exemplo, uma mesma atividade pode ser processada durante 3 unidades de tempo, consumindo 2 recursos renováveis e 5 recursos não renováveis, ou, de modo distinto, durar 6 unidades de tempo, e consumir 1 recurso renovável e 2 recursos não renováveis.

Apesar dos avanços dos métodos exatos para este problema, os tempos computacionais para estes algoritmos podem ser bastante excessivos. Nos trabalhos mais recentes, avaliados neste trabalho, são utilizadas Metaheurísticas na busca de soluções ótimas, tais como Algoritmo Genético, Busca Tabu, Simulated Annealing, entre outras.

Este trabalho teve origem no interesse de abordar o RCPSP com vistas à Gerência de Projetos, que é a aplicação de um conjunto de conhecimentos, habilidades e técnicas, na elaboração de atividades relacionadas com o objetivo de atingir um conjunto de metas definidas dentro de parâmetros de qualidade determinados, obedecendo a um planejamento prévio de prazos e custos. De acordo com o PMBOK® Guide, guia publicado pelo Project Management Institute, um projeto pode ser definido como uma seqüência de atividades (ou eventos) com início e fim definidos, executadas segundo uma ordem previamente determinada, objetivando alcançar uma meta temporal pré-estabelecida. Estas atividades possuem duração e precedência pré-determinadas. Além disso, lançam mão de recursos – escassos e limitados – que realizam as atividades.

A duração é um item crítico no planejamento de um projeto. Precisamos conhecer a duração do projeto a partir da rede de atividades associada e da duração estimada de cada atividade. O *Gerenciamento do Prazo do Projeto* é um dos processos na realização de um projeto, e é constituído por um conjunto de sub-processos que interagem com outras etapas do projeto: *Definição das Atividades*, *Seqüenciamento das Atividades*, *Estimativa da Duração das Atividades*, *Desenvolvimento do Cronograma*, e *Controle do Cronograma*. No *Seqüenciamento*, temos como objetivo a identificação e a documentação das interdependências entre as atividades, que podem ser essenciais, arbitradas ou impostas. As essenciais dizem respeito ao produto a ser gerado pelo projeto (por exemplo, o layout da planta a ser construída, ou as interfaces de subsistemas num projeto de software). As arbitradas (ou discricionárias) são aquelas definidas pela equipe de gerência do projeto, com base no conhecimento de “Best Practices”, particularidades do projeto, ou conforme alguma lógica específica. Por último, as impostas, também conhecidas por mandatórias ou de lógica rígida, que são inerentes à natureza do trabalho, envolvendo limitações físicas, como por exemplo preparar a fundação antes de levantar a estrutura, ou construir protótipo antes de testar o software. Estas dependências também se referem aos momentos de início e de fim das atividades: podem ser FS (Finish to Start) onde a atividade B inicia após o término da

atividade A; SS (Start to Start) onde B inicia quando A inicia; FF (Finish to Finish) onde B termina quando A termina; e SF (Start do Finish) onde B termina quando A inicia. Ainda conforme o PMBOK® Guide, o principal produto gerado ao final do Seqüenciamento das Atividades é o Diagrama de Rede do Projeto.

O processo de Desenvolvimento do Cronograma tem como entrada o Diagrama de Rede do Projeto, as estimativas de duração das atividades, as necessidades de recursos, calendários previstos, as restrições e premissas, o plano de gerenciamento de risco, e os atributos das atividades. A partir daí, lançando mão de ferramentas adequadas, é possível elaborar o cronograma do projeto, que define as datas iniciais e finais de cada tarefa.

As atividades, considerando-se a interdependência, devem ser dispostas em um Gráfico de Gannt. Pode-se descrevê-lo como sendo um gráfico plano de barras horizontais, onde cada barra representa uma atividade do projeto. Na direção horizontal encontram-se as durações em escala de tempo, e na direção vertical encontram-se as identificações das atividades. Quando se tratam de projetos longos e com grande número de atividades, este tipo de representação torna-se confuso, devido à falta de clareza na visualização dos inter-relacionamentos das atividades e na visão de conjunto. As figuras 1 e 2 representam, respectivamente, uma rede de um projeto, com as precedências e as demandas de recursos, e o gráfico de Gannt de uma possível solução. Na figura 1, temos as atividades (representadas pelos círculos numerados), as restrições de precedência (indicadas pelas setas), e as durações e as restrições de recursos indicados pela frações próximas a cada círculo de atividade: no numerador encontra-se a duração e no denominador a quantidade de recursos requeridos do tipo k (k=1), sendo o máximo disponível, em cada instante, igual a 4. A solução respectiva representada no gráfico de Gannt da figura 2 é resultante da aplicação do processo SGS Paralelo, detalhado na seção 2.1.1, e os retângulos em cor cinza indicam os períodos de tempo em que as atividades foram escalonadas: a atividade 2, sem antecessor, iniciou no instante 0 e terminou no instante 3, durando 4 instantes de tempo e utilizando 2 recursos; a atividade 4, iniciou ao término da predecessora (atividade 2) no instante 4 e terminou no instante 5, durando 2 instantes de tempo e utilizando 2 recursos; e assim por diante.

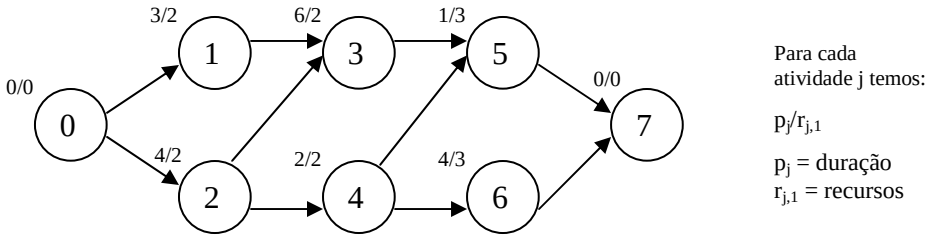

Figura 1 - Rede de um projeto, com as precedências e as demandas de recursos

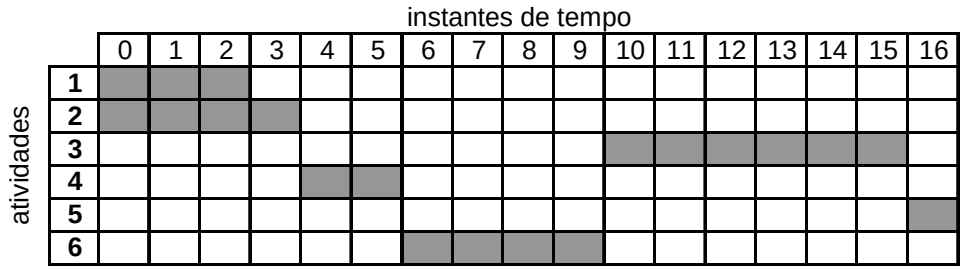

Figura 2 - Gráfico de Gannt de uma possível solução à rede de projeto da figura 1

Ainda de acordo com o PMBOK® Guide, existem técnicas para o desenvolvimento do cronograma, baseadas em análises matemáticas (CPM - Método do Caminho Crítico , PERT - Program Evaluation and Review Technique), em compressão da duração (Crashing, Fast Tracking), em simulações (Análise Monte Carlo), e em heurísticas (Critical Chain, SGS, metaheurísticas), que são o foco deste trabalho. Adicionalmente, existem softwares que auxiliam os Gerentes de Projetos nesta tarefa, como o MS-Project, que utiliza a heurística "*single pass methods*" – SGS –, o Primavera, o CS Project, o Fast Track Schedule, entre outros.

Neste trabalho propomos uma solução utilizando a Busca Tabu conjugada com Reconexão de Caminhos. Compõem este trabalho um estudo de construções de soluções utilizando Heurísticas, apresentadas no capítulo 2, uma heurística proposta para resolução deste problema, apresentada no capitulo 3, os resultados computacionais no capítulo 4 e, ao final, uma conclusão do trabalho no capítulo 5.

2. Heurísticas para solução do problema

Em um problema de otimização, deseja-se minimizar ou maximizar uma função de uma ou mais variáveis, as quais devem satisfazer a um conjunto de restrições que definem o domínio das soluções viáveis. As heurísticas são algoritmos capazes de encontrar soluções viáveis e, apesar de nem sempre garantirem o valor ótimo, podem vir a alcançar valores próximos a ele. As heurísticas podem ser classificadas em dois grandes grupos: as construtivas e as de melhoria ou de busca local. As heurísticas construtivas são aquelas onde, a partir de uma solução vazia, o algoritmo vai iterativamente escolhendo e adicionando um novo elemento à solução corrente até que esta se torne viável. Já as heurísticas de melhoria (ou busca local) são aquelas em que, partindo de uma solução inicial, o algoritmo vai efetuando pequenas mudanças locais na solução corrente, migrando então para outra solução, até que um determinado critério de parada seja atingido. Em uma heurística de busca local clássica, o critério de parada é atingido quando a solução corrente é tal que nenhum movimento possa ser realizado sem que se chegue a uma solução vizinha de custo pior ou igual ao da solução corrente. Neste caso, diz-se que a solução corrente é um mínimo local (Glover, 1989*).*

Algoritmos exatos obtêm solução livre de arredondamento em um tempo (polinomial ou não) que é função do tamanho da instância de entrada, e que pode tornar-se muito extenso e inaceitável a partir instâncias relativamente maiores. Para o problema do RCPSP existem algoritmos exatos (livres de erros de arredondamento) que, entretanto, suportam apenas pequenas instâncias (Coelho e Tavares, 2002). Já no campo das heurísticas, para a montagem de um cronograma respeitando-se a precedência entre as atividades, temos as Técnicas de Rede (PMBOK® Guide): o CPM – Método do Caminho Crítico – e o PERT (Program Evaluation and Review Technique) – Técnica de Avaliação e Controle de Projetos – semelhante ao CPM, mas que confere tratamento probabilístico às durações, com base em datas pessimistas, prováveis e otimistas. Estas técnicas surgiram no fim da década de 50. Entretanto, não se aplicam ao RCPSP por considerarem os recursos como infinitos, sem restrições de quantidade ou tipo. Para estas restrições, devemos contemplar outras técnicas, de maior robustez e abrangência, conforme apresentado a seguir.

2.1. SGS

Schedule Generation Schemes é uma heurística construtiva que gera (Kolisch e Hartmann, 1998), passo a passo, uma solução possível, estabelecendo esquemas parciais no tempo. Existem dois tipos de SGS disponíveis: o SGS Serial – que constrói a solução incrementando e seqüenciando as atividades elegíveis (que possuem recursos disponíveis e atividades precedentes concluídas) – e o SGS paralelo – que constrói a solução incrementando o tempo e avaliando todas as atividades elegíveis de iniciar em cada próximo instante de tempo.

O SGS Serial consiste de n estágios (n = número de atividades), onde em cada um é selecionada uma atividade entre as elegíveis e incluída na solução em construção, sendo programada o mais cedo possível, respeitando as restrições de recursos, e assim estabelecido o instante de término, estimando a execução da atividade sem preempção. Desta forma, as atividades são integralmente executadas em intervalos de tempos contíguos, com início o mais cedo possível e respeitando restrições de precedência e de recursos. O conjunto de elegíveis é composto pelas atividades pendentes que já tiveram todos seus predecessores escalonados.

O SGS Paralelo consiste de iterações em que o tempo é incrementado até o menor instante de tempo correspondente ao fim das atividades ativas (ainda não concluídas naquele instante de tempo considerado) e, correspondente ao instante assim determinado, são selecionadas todas as atividades elegíveis de serem iniciadas, considerando a precedência e a disponibilidade de recursos. Este método constrói soluções “non-delay”, ou seja, mesmo que ocorra preempção das atividades, não admite antecipação de qualquer sub-atividade, sem que isso implique em adiamento de outra atividade.

Ambos (serial e paralelo) não admitem antecipação de qualquer atividade, sem que isso implique em adiamento de uma outra.

A partir da precedência estabelecida na rede de projeto da figura 1, temos, a seguir, os exemplos de SGS Serial e Paralelo nas tabelas 1 e 2 [24] a seguir.

g	1	2	3	4	5	6
t_g	0	0	4	6	10	16
D_g	{1,2}	{2}	{3,4}	{3,6}	{3}	{5}
j	1	2	4	6	3	5

Tabela 1: Exemplo de SGS Serial

g	1	2	3	4	5	6
t_g	0	3	4	6	10	14
D_g	{1,2}	{ }	{3,4}	{6}	{5,6}	{5}
j	1,2		4,3		6	5

Tabela 2: Exemplo de SGS Paralelo

Figura 3: Exemplo de SGS Paralelo

Na tabela 1, correspondente ao SGS Serial, a primeira linha indica o estágio g do esquema de geração, a segunda refere-se ao instante t_g correspondente ao instante de início obtido pelo algoritmo para a atividade, a terceira linha apresenta o conjunto de atividades elegíveis D_g no estágio g, e a quarta indica a atividade j escolhida para início em t_g. Inicialmente, no estágio 1, sendo D_1 formado pelas atividades 1 e 2, seleciona-se aleatoriamente a atividade 1 a ser programada o mais cedo possível, ou seja, no instante 0, terminando no instante 2. No estágio 2, D_2 é formado apenas por 2, que é a única com predecessores já escalonados, sendo também escalonada no instante 0 e terminando no instante 3. No estágio 3, em que $D_3 = \{3,4\}$, ainda aleatoriamente, a atividade 4 é selecionada, sendo programada no instante 4, ao término da predecessora 2, e terminada no instante 5. No estágio 4, em que $D_4 = \{3,6\}$, é selecionada a atividade 6, que tem seu início mais cedo no instante 6, se estendendo até o instante 9. No estágio 5, apenas a atividade 3 é elegível, por restrições de precedência, e então $D_5 = \{3\}$. Como a atividade 3 consome apenas dois recursos dos quatro disponíveis, porém dura 6 instantes consecutivos, ela é iniciada somente no instante 10 e termina no instante 15. Por fim, no

estágio 6, temos apenas a atividade 5 em D_6, que é iniciada e finalizada no instante 16. Desta forma, conclui-se um makespan de 17 unidades de tempo.

Na tabela 2, correspondente ao SGS Paralelo, a primeira linha indica o estágio g do esquema de geração, a segunda refere-se ao instante t_g correspondente ao início da atividade escolhida, a terceira linha apresenta o conjunto de atividades elegíveis D_g no estágio g, e a quarta indica a atividade j escolhida para início em t_g. Inicialmente, no estágio 1, temos $t_1 = 0$ (instante zero), e o conjunto de atividades elegíveis D_1 composto pelas atividades 1 e 2. Aleatoriamente, a atividade 2 é escolhida. Como utiliza somente dois recursos, a atividade 1 também é escalonada neste instante, por consumir os demais dois recursos disponíveis. Assim as atividades 1 e 2 terminam, respectivamente, nos instantes 2 e 3. Prosseguindo, no estágio 2 temos $t_2 = 3$, que é o instante mais recente em que uma atividade concluída liberou recursos (atividade 1). Porém D_2 é vazio, porque não existem, neste momento, atividades pendentes com precedessores já escalonados. No estágio 3, então, temos $t_3 = 4$, instante mais recente em que uma atividade concluída liberou recursos (atividade 2), e temos D_3 composto por 3 e 4. Aleatoriamente, a atividade 4 foi escolhida para iniciar em 4, utilizando dois recursos, sendo retirada de D_3. Como a atividade remanescente em D_3 é a atividade 3, que por sua vez também consome apenas dois recursos, ainda disponíveis, ela também tem seu início programado para o instante $t_3 = 4$. Prosseguindo, no estágio 4 obtemos $t_4 = 6$, que corresponde ao instante em que atividade 4 libera dois recursos, e onde não há como escalonar a atividade elegível $D_4 = \{6\}$, devido às restrições de recursos, pois dois ainda estão sendo utilizados na atividade 3. O algoritmo prossegue para o estágio 5, onde $t_5 = 10$ (quando a atividade 3 liberou recursos) e D_5 é formado por apenas pelas atividades 5 e 6. A atividade 6 é escolhida aleatoriamente e termina no instante 13. Finalmente, no estágio 6 temos $t_6 = 14$ (instante em que a atividade 6 liberou recursos) e $D_6 = \{5\}$. A atividade 5 é, então, iniciada e terminada no instante 14, concluindo o algoritmo, com makespan igual a 15 unidades de tempo. O resultado está representado na figura 3.

Ambos algoritmos geram soluções ótimas para problemas sem restrição de recursos e a principal diferença entre eles ocorre na construção do algoritmo.

2.2. X-Pass

Os métodos X-Pass (Kolisch e Hartmann, 1998) são heurísticas construtivas, e combinam regras de prioridade com métodos SGS (Serial e Paralelo) para construir uma ou mais soluções, e são classificados em Single-Pass ou Multi-Pass. A regra de prioridade é usada para selecionar uma atividade no conjunto de elegíveis. Como exemplos, podemos citar: MTS (Most Total Successors – maior quantidade de sucessores diretos ou induzidos), LFT (latest finish time – maior tempo de conclusão), LST (latest start time – maior tempo de início), WCS (worst case slack – pior caso de folga se a atividade não for escolhida entre as elegíveis), SPT (shortest processing time – menor tempo de processamento) e RAN (seleção aleatória).

2.2.1. Single-Pass

É a heurística mais antiga do método X-Pass, e aplica um único SGS (Serial ou Paralelo) na construção e uma única regra de prioridade na seleção das atividades elegíveis, para a elaboração de apenas uma solução possível.

2.2.2. Multi-Pass

Neste tipo de X-Pass, em lugar da utilização de apenas uma regra de prioridade e de uma única passagem, são realizadas diversas passagens simples (Single Pass), com diferentes regras de prioridade, para a escolha do melhor resultado. Há diversas possibilidades de se combinar SGS (Serial ou Paralelo) e regras de prioridade em um método Multi-Pass. Pode ser classificado em: Múltiplas Regras de Prioridade, Escalonamento com Avanço-Recuo (Forward-Backward Scheduling), e Métodos de Amostragem.

No método de Múltiplas Regras de Prioridade, o SGS é aplicado diversas vezes, cada uma com uma regra de prioridade distinta, na escolha das atividades elegíveis.

No Escalonamento com Avanço-Recuo (Forward-Backward Scheduling), aplicam-se as regras de prioridade como em Single-Pass. O recuo corresponde à aplicação na ordem reversa das atividades. O avanço e o recuo são aplicados alternadamente até alcançar o melhor makespan.

Métodos de Amostragem geralmente usam um SGS (serial ou paralelo) e uma regra de prioridade, que indica a probabilidade de uma atividade j ser escolhida entre as elegíveis

D_i. Soluções diferentes são obtidas por diferentes modos de cálculo das prioridades: amostragem aleatória (random sampling) – probabilidades iguais entre as atividades, amostragem aleatória tendenciosa (biased random sampling) – os valores das prioridades indicam diretamente a probabilidade correspondente, e, por fim, a amostragem aleatória tendenciosa baseada em desvio (regret based biased random sampling) – utiliza os valores de prioridade indiretamente, através dos valores de desvio (regressão), ou seja, a probabilidade da atividade j ser selecionada é função de um parâmetro de desvio, que por sua vez depende de considerações sobre o conjunto de decisão D_i.

2.3. Metaheurísticas

Heurística pode ser definida como sendo uma técnica que procura boas soluções (próximas da otimalidade) a um custo computacional razoável, sem, no entanto, estar capacitada a garantir a otimalidade, bem como garantir quão próxima uma determinada solução está da solução ótima. A grande desvantagem das heurísticas reside na dificuldade de fugir de ótimos locais. As Metaheurísticas foram desenvolvidas para possibilitar a saída destes ótimos locais, permitindo a busca em regiões mais promissoras.–Elas utilizam a combinação de heurísticas construtivas e de busca local procurando obter resultados.

Dentre as metaheurísticas que surgiram ao longo das últimas décadas, destacam-se: Algoritmos Genéticos (AGs) (Goldberg, 1989), Simulated Annealing (Kirkpatrick, 1983), Busca Tabu (TS) (Glover, 1986), Greedy Randomized Adaptative Search Procedure (GRASP) (Feo & Resende, 1995), Colônia de Formigas (Taillard, 1999), Variable Neighborhood Search (VNS) (Mladenovic & Hansen, 1997), entre outros.

2.4. Metaheuristicas para o RCPSP

Ao longo do tempo, surgiram variadas abordagens para resolução do RCPSP. O modelo clássico do problema foi abordado por Kolisch e Hartmann, que elaboraram resumos reunindo implementações de diversos autores, desenvolvidos com o foco no RCPSP (tabela 3) e utilizando metaheurísticas conhecidas. Dentre eles, detalhamos, mais adiante, o trabalho de Baar, Brucker e Knust (1997), que apresentaram duas soluções com Busca Tabu. Também identificamos os trabalhos de Merkle et al (2000) – usando

Colônia de Formigas – , Valls et al (2001)– usando uma abordagem evolutiva – , e Hartmann (2002)– utilizando Algoritmo Genético Adaptativo.

Em abordagens mais recentes, Nonobe e Ibaraki (2003) – usando Busca Tabu –, Bouleimen e Lecocq (2003) – reutilizando o Simulated Annealing de seu trabalho anterior de 1998 –, Fleszar e Hindi (2004) – usando VNS (Variable Neighbourhood Search) –, e Brucker e Knust (2003) – usando Lower Bounds –, apresentaram estudos sobre uma versão mais abrangente do problema, incluindo recursos renováveis e não-renováveis, multi-modos e preempção das atividades, esta última incluída somente no trabalho de Nonobe e Ibaraki. Estas características são descritas no item 3.2.

Outra abordagem, intitulada Labor Constrained Scheduling Problem, onde as atividades são aplicadas por recursos restritos em um conjunto de máquinas, agrupadas em ordens e com precedência entre elas, foi avaliada nos trabalhos de C. C. B. Cavalcante, C. C. Ribeiro, V. C. Cavalcante e C. C. de Souza (2000), que aplicam metaheurísticas paralelas cooperativas, no de C. C. B. Cavalcante e C. C. de Souza (1997), utilizando Busca Tabu, assim como por C. C. Souza e L. A. Wolsey (1997), utilizando Programação Inteira.

Também foram observadas outras apresentações ou soluções do problema, bastante distintas das demais. O PCDR – Problema de Custo de Disponibilidade de Recursos – , usando a metaheurística Scatter Search, apresentado por Yamashita (2003), considera o tempo escasso e os recursos ilimitados, regidos por uma função não-decrescente de custo. Nonobe e Ibaraki (2001) consideraram que o início mais cedo ou tardio de uma atividade infere em custo (time lag cost). O uso de uma metaheurística híbrida Scatter Search / Eletromagnetismo, foi apresentada por D. Debels, B. Reyck, R. Leus e M. Vanhoucke (2003). Valls et al (2003) apresentaram uma nova heurística, com base na reordenação das atividades críticas.

Artigo	Metaheuristica	Representação	SGS	Operador
Baar et al. (1997)	Busca Tabu	lista de atividades	s	shift cam critico
		esquema escalonado	relacionado	mov relacionados
Bouleimen, Lecocq (1998)	Sim. Annealing	lista de atividades	s	shit
Cho, Kim (1997)	Sim. Annealing	chave randomica	mod p	pairw int
Hartmann (1998)	Alg. Genetico	lista de atividades	s	crossover 2 pts
		chave randomica	s	crossover 2 pts
		regra de prioridade	s	crossover 2 pts
Kohlmorgen et al. (1998)	Alg. Genetico	chave randomica	s	crossover 2 pts
Lee, Kim (1996)	Sim. Annealing	chave randomica	p	pairw int
	Busca Tabu	chave randomica	p	pairw int
	Alg. Genetico	chave randomica	p	crossover 1 pt
Leon, Ramamoorthy (1995)	Alg. Genetico	chave randomica	mod p	espaço problema
	FFS	chave randomica	mod p	espaço problema
	BFS	chave randomica	mod p	crossover 1 pt
Naphade et al. (1997)	BFS	chave randomica	mod p	espaço problema
Pinson et al. (1994)	Busca Tabu	lista de atividades	s	adj pairw int
		lista de atividades	s	pairw int
		lista de atividades	s	shift
Sampson, Weiss (1993)	Sim. Annealing	shift vector	rec ext	mov relacionados

Tabela 3: soluções do RCPSP clássico, relacionadas por Kolisch e Hartmann (1998 e 1999).

Nas soluções apresentadas pelos diversos autores e incluídas nas pesquisas de Kolisch e Hartmann, elencadas na tabela 3, percebemos não só a grande variedade de possibilidades já avaliadas, como a diversidade de novas combinações possíveis. Para cada Metaheuristica utilizada poderemos, por exemplo, inserir variações nos algoritmos, nas representações dos esquemas de construção das soluções e nos operadores utilizados para buscas nas vizinhanças. Na terceira coluna da tabela temos o tipo de representação adotado para o esquema de construção da solução: lista de atividades (as atividades são posicionadas numa lista ordenada, respeitando as precedências entre elas), chave randômica (as atividades são posicionadas num vetor, e a cada atividade é associado um número, similar ao valor de prioridade), lista de prioridades (as regras de prioridades são dispostas numa lista ordenada, e cada regra corresponde a uma atividade a ser escalonada), shift-vector (utiliza um vetor, onde cada valor corresponde a um acréscimo a ser adicionado ao instante de início da atividade correspondente). Uma última representação é o Esquema Escalonado (schedule scheme). Ele é constituído por quatro relações disjuntas do par ordenado de atividades (i,j): relação C (conjunções), onde i é restrição de precedência de j (isto é, i→j); relação D (disjunções), onde i e j não podem

ocorrer paralelamente em nenhum momento; relação N (paralelismo), onde i e j devem ser processadas em paralelo; e a relação F, onde i e j não têm qualquer restrição (flexibilidade). A quarta coluna da tabela indica se o SGS utilizado foi serial, paralelo, mod-p (paralelo modificado, onde a inclusão de uma atividade escolhida entre as elegíveis não ocorre necessariamente no menor instante inicial, permitindo um atraso), relacionado (constituído de um SGS adaptado, onde as atividades são escalonadas num procedimento intitulado List-Schedule), e rec ext (um SGS estendido, onde a escolha de uma atividade elegível corresponde a uma regra de prioridade). A quinta coluna indica os operadores utilizados na metaheurística que serão detalhados mais adiante.

Baar et al. (1997), desenvolveram dois algoritmos Tabu. No primeiro, utilizaram a representação de lista de atividades com SGS serial, e a vizinhança foi definida por três tipos de movimentos com base no caminho crítico, que é o conjunto de atividades que, sozinhas, determinam a duração total do projeto. No segundo, utilizaram a representação do esquema escalonado. Em ambos foram usadas listas tabu dinâmicas, assim como heurísticas de inicialização com base em regras de prioridade, ou seja, para obter uma boa solução inicial, aplicaram diferentes heurísticas baseadas em regras de prioridade e selecionaram a melhor solução encontrada.

Bouleimen e Lecocq (1998) propuseram um algoritmo Simulated Annealing com representação do problema através de uma fila de atividades, construída na solução inicial através de uma regra de prioridade determinada, um SGS serial para construção do esquema solução e um operador *shift* nos movimentos pela vizinhança, transferindo uma determinada atividade j_s para a posição imediatamente anterior a uma outra atividade j_q.

Hartmann (1998) propôs um Algoritmo Genético baseado na representação de lista de atividades, e comparou com outros AG's utilizando chave randômica e regras de prioridade. Todos utilizaram SGS serial e operador *crossover* de 2 pontos. Na lista de atividades, a população inicial foi formada com o uso de amostragem aleatória tendenciosa baseada em desvio (regret based biased random sampling method) e regra de prioridade LFT (Latest Finish Time).

Cho e Kim (1997) desenvolveram um algoritmo Simulated Annealing modificado sobre o trabalho de Lee e Kim (1996), estendendo a representação de chave randômica de modo a permitir o atraso nas atividades elegíveis em um método SGS paralelo adaptado,

ou seja, diferentemente do método padrão, em que as atividades são sempre escalonadas o mais cedo possível, são permitidos atrasos em alguns casos, com base nas prioridades do vetor de chave randômica, com o objetivo de ampliar o espaço de busca.

Lee e Kim (1996) propuseram 3 abordagens: Simulated Annealing, Algoritmo Genético e Busca Tabu. Todas baseadas em chave randômica, com SGS paralelo. O Simulated Annealing e a Busca Tabu utilizam uma versão restrita do movimento *pairwise interchange*, que é a troca de pares de atividades pertencentes a uma lista de atividades, de modo a permitir que a solução associada respeite as restrições de precedência. Como um caso especial, atividades adjacentes j_q e j_{q+1} podem trocar de posição, mesmo não guardando precedência entre elas. O Algoritmo Genético utilizou operador *crossover* de um ponto.

Kohlmorgen et al (1998) desenvolveram um Algoritmo Genético com chave randômica e operador crossover de dois pontos e testaram em computadores paralelos.

Leon e Ramamoorthy (1995) testaram abordagens FFS (First Fit Strategy) e BFS (Best Fit Strategy), ambas procedimentos de heurística gulosa, assim como Algoritmo Genético. A FFS aceita imediatamente uma solução vizinha que melhore a corrente, rejeitando aquelas que deterioram. A BFS se assemelha à Busca Tabu, pois procura a solução numa vizinhança e aceita a melhor encontrada, até que nenhuma outra melhore a corrente. Todas as abordagens foram baseadas em representação de chave randômica, na sua versão baseada em espaço-problema, que é outro tipo de representação similar à chave randômica, onde as chaves randômicas são inicializadas com valores calculados por uma regra de prioridade (em lugar de valores aleatórios), e um SGS paralelo modificado é usado para a montagem das soluções. O operador unário é definido pela vizinhança baseda no espaço-problema, por exemplo um *pairwise interchange*, e o binário é o *crossover* de um ponto.

3. Heurística Proposta

O problema descrito (Nonobe e Hibaraki, 2000) como se segue, compõe-se de determinados conceitos de recursos e atividades.

Seja R um conjunto de recursos e J um conjunto de atividades. O conjunto R é dividido em um conjunto de recursos renováveis R^{re} e um conjunto de recursos não renováveis R^{non}. Recursos renováveis são aqueles onde o limite da quantidade disponível é restrito somente a um determinado período de tempo e é independente dos demais períodos, enquanto que os recursos não renováveis apresentam limites de quantidades referentes à totalidade de tempo do projeto, sem restrições em cada período. Por exemplo, mão-de-obra e máquinas são consideradas recursos renováveis, enquanto que matérias-primas e verba orçamentária são exemplos típicos de recursos não renováveis.

Cada atividade $j \in J$ deve ser processada de um modo m_j escolhido a partir de um conjunto M_j. Para cada modo $m_j \in M_j$, são previamente especificados: o tempo de processamento p_{mj}, o conjunto de recursos $R_{mj} = R^{re}_{mj} \cup R^{non}_{mj}$ e seus correspondentes requisitos de quantidade. Uma vez iniciado o processamento de uma atividade j no modo m_j, ela é processada nos p_{mj} períodos de tempo consecutivos subseqüentes, até o término.

Uma solução é representada por um esquema (m,s) constituído por duas listas, $m = (m_j \mid j \in J)$ e $s = (s_j \mid j \in J)$, onde m_j é o modo em que a atividade j é processada e s_j é o instante de início de processamento da atividade j. O instante de conclusão da atividade é dado por $c_j = s_j + p_{mj}$. Os valores de p_{mj}, s_j e c_j são presumidos inteiros não negativos. Alternativamente, uma solução pode ser representada por um esquema (m,l), sendo $l = (j \mid j \in J)$ uma lista das atividades, cuja ordenação corresponde à ordem de escalonamento destas mesmas atividades.

As restrições de recursos são classificadas em dois tipos. Restrições de recursos renováveis e de recursos não renováveis. Para cada recurso do tipo k renovável $r^{re}_k \in R^{re}$, em cada instante de tempo (t-1, t] (t=1,2,...), a quantidade total requerida de r^{re}_k por todas as atividades simultâneas naquele instante t não pode exceder o total disponível de R^{re}_k. Para cada recurso não renovável do tipo k, ou seja $r^{non}_k \in R^{non}$, a quantidade total, requerida no decorrer de todo o projeto, não pode exceder a quantidade disponível previamente especificada R^{non}_k . A quantidade de recurso requerida por uma atividade

depende apenas do modo e independe do instante de início. Assim, estas restrições impedem algumas combinações de modos.

Existem também restrições de precedência simbolizadas por i < j que indicam que j não pode iniciar antes que i tenha terminado, isto é, $c_i \leq s_j$, e $i \in \boldsymbol{P}_j$, conjunto de atividades predecessoras a j. Considerando-se uma atividade denominada *sink* como a que sucede todas as demais do projeto por restrições de precedência, e cujo tempo de processamento é zero, o objetivo de minimizar o tempo total (*makespan*) do projeto pode ser traduzido pela minimização da função $c_{sink} \geq 0$, como mostrado em (1) a seguir, sujeito às restrições de precedência (2), às restrições de recursos renováveis do tipo k em cada instante de tempo (3) e às restrições de recursos não renováveis do tipo k, referente ao modo m_j escolhido para execução de cada atividade j (4).

$$\text{Min} \quad c_{sink} \geq 0 \qquad (1)$$

$$s_i + p_{mi} \leq s_j \qquad , i \in \boldsymbol{P}_j \text{ , executada no modo } m_i \qquad (2)$$

$$\sum r^{re}_{k,mj} \leq R^{re}_k \qquad , j \in \boldsymbol{A}_t \text{, executada no modo } m_j \qquad (3)$$

$$\sum r^{non}_{k,mj} \leq R^{non}_k \qquad , j \in J, \text{ executada no modo } m_j \qquad (4)$$

, $\boldsymbol{P}_j$ = conjunto de atividades predecessoras diretas à atividade j

, $\boldsymbol{A}_t$ = conjunto de atividades em execução no instante t

Apoiando-se numa combinação dos trabalhos de Nonobe e Hibaraki [35][34], descritos no item 3.2, foi desenvolvida, neste trabalho, uma solução para o problema, utilizando a metaheuristica Busca Tabu, acrescida de intensificação proporcionada pela Reconexão de Caminhos, ambos descritos no item 3.1. No item 3.3 a implementação é descrita detalhadamente.

3.1. Metaheurísticas utilizadas

O trabalho foi baseado na Busca Tabu, intensificada pelo uso da Reconexão de Caminhos. A escolha recaiu sobre esta heurística pelas possibilidades proporcionadas pelo algoritmo e pelos resultados promissores demonstrados em diversos trabalhos [1][5][10][34].

3.1.1. Busca Tabu (BT)

Introduzido por Fred Glover (Universidade de Colorado – Boulder) e Pierre Hansen (Universidade de Montreal) em 1986, considera que, para melhorar a eficiência num processo de exploração, deve-se não somente guardar as informações locais, mas também armazenar informações relacionadas ao processo de exploração como um todo. Esta sistemática implica no uso de memórias, característica da BT. Enquanto outros métodos guardam apenas a melhor solução obtida até o momento, ele armazena informações sobre o itinerário utilizado (memória adaptativa), junto com as soluções mais recentemente visitadas. Estas informações servem para guiar os próximos movimentos.

Entre os recursos que utiliza, temos: lista tabu, aspiração, memórias de curto e longo prazo, intensificação, diversificação, lista de candidatos, contadores de freqüência. A idéia é permitir que haja movimento para soluções vizinhas mesmo que essas eventualmente deteriorem o valor da função objetivo. Tipicamente, o movimento é feito de modo que se chegue à solução vizinha de melhor custo mesmo que esta seja pior do que a solução corrente. Contudo, no caso de piora no custo, haveria um risco de que o movimento inverso levasse o algoritmo de volta à solução corrente da iteração anterior. A maneira encontrada para impedir este comportamento cíclico na Busca Tabu foi criar uma lista de movimentos que permanecem proibidos ou tabu durante um certo número de iterações. Este número de iterações é denominado de prazo tabu. O critério de parada usualmente adotado restringe-se a um simples limite sobre o número total de iterações ou então sobre o número total de iterações sem que o custo da melhor solução visitada tenha sido alterado.

A partir de uma solução inicial, o método procura melhorar a solução corrente (busca local), efetua buscas numa vizinhança de um ótimo local (intensificação, usando memória de curto prazo), busca em regiões promissoras pouco ou ainda não exploradas (diversificação), evita retornar por caminhos recentemente trilhados com base nos movimentos proibidos incluídos na lista Tabu, onde também aplica critérios para a retirada de movimentos da lista Tabu e a reintegração deles ao processo (aspiração).

Alguns parâmetros, que precisam ser determinados, definem o critério de parada, o prazo tabu e o tamanho da lista tabu. O ajuste destes parâmetros é uma tarefa essencial

para o bom desempenho do algoritmo. A atualização da lista tabu, no método estático, é realizada com base em um determinado número de iterações e no tamanho da lista (Tabu Tenure ou Prazo Tabu). Para reduzir as chances de retornar ao ótimo local, a manutenção da lista pode ser realizada pelo método dinâmico, onde a permanência e o tamanho da lista variam aleatória ou sistematicamente, de acordo com os movimentos considerados.

Cabe observar que a Busca Tabu, apesar da tentativa de escapar dos mínimos locais, não oferece qualquer garantia de que irá produzir uma solução ótima. Contudo, em geral, o que se verifica é que as soluções geradas são bem superiores àquelas retornadas pelas heurísticas clássicas de busca local [8].

3.1.2. Reconexão de Caminhos (RC)

A técnica de Reconexão de Caminhos foi proposta por Glover (1996) e pode ser vista como um procedimento para melhorar a solução corrente, explorando trajetórias entre soluções elite encontradas no decorrer de algum algoritmo de busca (Glover, Laguna, 1997). Uma “solução elite” é aquela que apresentou melhor resultado que quaisquer das outras precedentes, obtidas durante uma busca (Vogt, 2005). Em geral, um algoritmo de busca local (Busca Tabu e outros) armazena informações sobre o caminho entre soluções contíguas. O espaço solução entre duas soluções consecutivas pode ser longo e diversificado. A idéia da Reconexão de Caminhos é buscar um caminho mais direto entre essas soluções consecutivas.

Nesta procura por soluções melhores, incorpora atributos das soluções de boa qualidade até então encontradas. Pode ser interpretado como um método evolucionário, pela combinação de elementos de outras soluções [19]. Diferente de outros procedimentos evolucionários, em lugar de processos randômicos, como no Algoritmo Genético, utiliza regras sistemáticas e determinísticas para combinar soluções. Gera novas soluções explorando trajetórias que conectam pares de soluções elite: começa pela que chama de *solução inicial*, e prossegue avaliando soluções intermediárias no caminho que liga à solução elite seguinte, chamada de *solução guia*. A partir da *solução inicial* são geradas soluções intermediárias que incorporam, uma após outra, o conjunto de modificações que diferenciam a *solução inicial* da *solução guia*, terminando por igualar a solução corrente à *solução guia*.

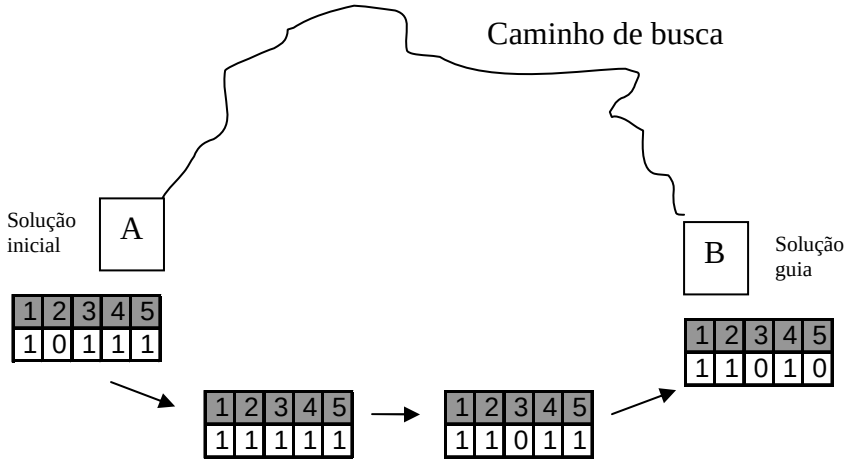

Figura 4 – Reconexão de Caminhos

A figura 3 ilustra uma Reconexão de Caminhos [45], entre os pontos A e B, que representam soluções elite, encontradas consecutivamente no caminho de busca do algoritmo de busca local. Sendo a solução representada por um vetor de 5 posições, com cada posição assumindo valores 0 ou 1, a *solução inicial* A é representada por (1,0,1,1,1) e a *solução guia* B por (1,1,0,1,0). O método percorre um caminho mais direto entre os pontos A e B do que o traçado pelo caminho de busca. Para este novo trajeto, o algoritmo primeiro identifica as diferenças entre a solução inicial e a solução guia, ou, como no exemplo, as posições 2, 3 e 5 do vetor representativo da solução. Então prossegue, evoluindo as trocas, uma a uma, executando seqüencial e prioritariamente aquelas que apresentam melhor resultado. Assim, comparando os resultados obtidos pelas três trocas (posições 2, 3, e 5) e escolhendo, no exemplo, a posição 2 (a de melhor resultado), procede-se com a troca do conteúdo da solução inicial na posição 2 (valor 0) para o correspondente na solução guia (valor 1), gerando uma nova solução (1,1,1,1,1). Prossegue-se, a partir daí, selecionando outra troca entre as remanescentes (posições 3 e 5) pelo critério de melhor resultado. No exemplo, foi modificada a posição 3 (valor 1) para o correspondente na solução guia (valor 0), resultando na nova solução (1,1,0,1,1). Por fim, considerando que a modificação da posição 5 corresponderá exatamente à solução guia B, o algoritmo é interrompido, prosseguindo para os pares seguintes de soluções elite. Assim, a Reconexão de

Caminhos pesquisa novas soluções sobre estes caminhos mais curtos entre as soluções elite da busca local, buscando melhores resultados.

A Reconexão de Caminhos possui 2 estratégias básicas [30][41]: de pós-otimização, aplicada sobre o conjunto de soluções elite obtido, ou de intensificação, a cada ótimo local, após a fase de busca local. Neste caso, é aplicada sobre a solução corrente resultante de uma busca local e sobre uma das soluções elite encontradas anteriormente pelo algoritmo de busca. Algumas alternativas vêm sendo consideradas e combinadas em implementações recentes [30][41], buscando-se alternativas para combinar tempo de processamento e qualidade de solução: aplicar a reconexão de caminhos periodicamente, e não a cada iteração, para desonerar o tempo total; usar duas trajetórias distintas para um mesmo par de soluções elite, trocando entre elas as posições de solução inicial e de solução guia; utilizar apenas uma trajetória de uma solução inicial para uma solução guia; interromper a trajetória antes de completá-la (reconexão de caminhos truncada).

3.2. Busca Tabu com Reconexão de Caminhos para o RCPSP

Foi desenvolvida uma solução utilizando-se uma Busca Tabu, baseada em dois artigos anteriores de Koji Nonobe e Toshihide Ibaraki, acrescida de uma intensificação proporcionada pelo uso da Reconexão de Caminhos.

Nonobe e Hibaraki (2003) [34]– NI-TPSPGen – propuseram uma solução Tabu para uma versão generalizada do RCPSP. Além das precedências, dos recursos renováveis / não renováveis e dos múltiplos modos, incluíram outras restrições: permitindo a preempção (atividades podem ser subdivididas) como relaxação, restringem as quantidades máximas de subdivisões, as quantidades máximas de subdivisões que podem ser executadas em paralelo, e, por fim, os tempos máximos de intervalo entre as atividades com relação de precedência. Desta proposta, destacamos a representação das soluções em pares de listas, chamados esquemas (**m**,**l**), onde **l** é a *lista de atividades* e representa uma relação ordenada das atividades, dispostas conforme a ordem de escalonamento destas atividades no cronograma do projeto, e **m** representa a *lista de modos*, que indica os modos em que cada atividade, daquela solução, será executada no projeto. Além destas características, também enfatizamos a navegação na vizinhança da

Busca Tabu, que utilizou dois movimentos: a *troca de modo* das atividades (ChangeMode), que troca o modo de execução de uma atividade na *lista de modos*, e a *troca de ordem* sobre a *lista de atividades* a serem escalonadas (ShiftForward), que, sobre um par $A,B \in J$, tal que A é anterior a B na ordem (de escalonamento) da *lista de atividades*, posiciona B imediatamente antes de A na mesma lista. A solução inicial foi gerada com base em duas construções distintas: para a construção da *lista de modos* foi utilizada uma Busca Tabu, proposta em [37] com a finalidade de solucionar um problema de restrições (CSP) genérico, e para a construção da *lista de atividades* foi utilizada a regra de prioridade denominada Most Total Successors, que confere precedência às atividades com maior quantidade de sucessores diretos ou indiretos.

No ano de 2000, também com uma abordagem de Busca Tabu, Nonobe e Ibaraki [35] – NI-TPSP – avaliaram uma versão mais tradicional do problema, incluindo o tratamento multimodo das atividades e a diversidade de recursos (renováveis e não renováveis), além das restrições de precedência entre as atividades. Adicionalmente, caracterizou-se um tipo particular de precedência, chamada de *precedência mediata*, onde não é admitido intervalo de tempo entre o instante de término da precedente e o instante de início da sucessora. Na movimentação pela vizinhança da Busca Tabu, foram utilizados três movimentos, em todos buscando limitar o espaço de busca: o ChangeMode, sobre a *lista de modos*, e dois outros sobre a *lista de atividades*, o ShiftAfter e ShiftBefore. O ChangeMode troca o modo de execução de uma atividade na *lista de modos*. O ShiftAfter e o ShiftBefore escolhem um par de atividades $A,B \in J$, tal que A é anterior a B na ordem (de escalonamento) da *lista de atividades*, e trocam suas posições nesta lista: o ShiftAfter posiciona A imediatamente após B, e ShiftBefore coloca B na posição imediatamente anterior a A. Também classificou as restrições em duras (hard), quando não podem ser flexibilizadas, como recursos renováveis e relações de precedência, e em flexíveis (soft), quando podem ser maleáveis, conferindo-se pesos às métricas de violação destas restrições, exemplificadas por recursos não renováveis. As soluções, também representadas em esquemas (m,l), eram construídas (escalonadas) com base em um algoritmo que chamaram de Construct, que estabelece o instante de início e de término das atividades, respeitando a ordem apresentada pela lista l, os modos de cada atividade indicados na lista m, e as restrições de recursos disponíveis.

Neste trabalho, optou-se por uma versão mais tradicional do RCPSP, onde, além das restrições de precedência e de recursos, incluiu-se a multiplicidade de modos na execução das atividades e de tipos de recursos – renováveis e não renováveis . Com base nos resultados já alcançado por outros autores, resultados estes reunidos na biblioteca de instâncias para o RCPSP, a PSPLIB (http://129.187.106.231/psplib/) e, especificamente, nos resultados obtidos pelo uso de Busca Tabu em NI-TPSPGen e em NI-TPSP, buscamos identificar a melhoria decorrente de uma intensificação através da Reconexão de Caminhos. A preempção de atividades foi descartada devido à inexistência de instâncias para este tipo de problema, além de apresentar menor aplicação em construção de cronogramas de projetos.

Ainda neste trabalho, como em NI-TPSPGen, estruturaram-se as soluções do problema em esquemas (m,l), isto é, pares de listas de modos e de listas de atividades, e utilizou-se a mesma formatação para a Busca Tabu. A solução inicial ($m^{(0)}$, $l^{(0)}$) corresponde à lista de modos $m^{(0)}$ viável, criada utilizando-se a Busca Tabu [37] aplicada com abordagem genérica sobre um CSP (Constraint Satisfaction Problem), e à lista de atividades $l^{(0)}$, gerada com base em uma regra de prioridade, a *Most Total Successors*. Novamente, a regra MTS prioriza as atividades com mais sucessores (diretos ou induzidos), pois estas ocasionam mais restrições [33]. Após a geração da solução inicial, é processada uma outra Busca Tabu, a principal, que varre a vizinhança N(m,l) em dois movimentos: o operador ChangeMode(j,m'$_j$), que muda o modo m_j, associado à atividade j, para m'$_j$, escolhido pelo critério de viabilidade e de menor consumo de recursos, e o operador ShiftForward(i,j), que seleciona a atividade j posicionada em u_j na lista de atividades l, e a reposiciona imediatamente antes da atividade i, esta posicionada em u_i, sendo $1<u_i<u_j<|J|$.

Do algoritmo NI-TPSP, utilizou-se, parcialmente, o procedimento Construct, semelhante ao algoritmo SGS, criado para construir as soluções viáveis a partir das soluções representadas pelos esquemas (m,l), respeitando as restrições de precedência e de recursos disponíveis a cada instante, e estabelecendo o instante de início e de término das atividades executadas na ordem da lista de atividades l e nos modos estabelecidos na lista de modos m. Foram ignorados os dispositivos para implementação do controle de

precedências imediatas, que fogem ao escopo do trabalho. Assemelham-se, em NI-TPSP, NI-TPSPGen e neste trabalho, a representação das soluções pelo esquema (m,l).

De acordo com a biblioteca de instâncias utilizada no problema, cada uma possui um horizonte máximo de tempo $T_{máx}$ para conclusão do projeto respectivo. Este limitador foi incluído na implementação, como uma restrição de tempo.

Com vistas a avaliar os tempos de processamento, buscou-se comparação com os resultados obtidos no trabalho de Nonobe e Ibaraki [35], que processaram cada instância durante 5000 iterações. Assim, foram estabelecidos 5 ciclos de 1000 iterações cada, sem prever limitação por iterações consecutivas sem melhoria.

Na solução desenvolvida, as restrições de recursos não renováveis foram consideradas duras (hard), assim como as de recursos renováveis e de precedência: assim, não foram admitidas soluções que apresentassem utilização excedente destes recursos, assim como aquelas que ultrapassassem o tempo máximo aceitável para o projeto (restrição de tempo).

Seguindo a estratégia de pós-otimização da RC, ao final de todas as iterações da busca Tabu, a implementação executa a Reconexão de Caminhos, buscando melhorar as soluções elite encontradas no decorrer do Tabu.

Com os recursos e dispositivos expostos acima, a estrutura básica da implementação (figura 4) é composta por 3 blocos principais: primeiro a geração da solução inicial (figura 6), em segundo as cinco iterações do conjunto de BT nas vizinhanças da lista de atividades (figura 7) e da lista de modos (figuras 8 e 9) e, em terceiro, a Reconexão de Caminhos (figura 10). Para a construção da solução, ou seja, o estabelecimento dos instantes de início (s_i) e de conclusão (c_i) de cada atividade i, a partir de um esquema (m,l), é utilizado o procedimento Construct (figura 5).

Estrutura da Implementação

1. Entrada: J, $\boldsymbol{P}_j$, M_j, R^{re}_k , R^{non}_k, p_{mj}, $r^{re}_{k,mj}$, $r^{non}_{k,mj}$, $T_{máx}$;
2.
3. {BT: Solução Inicial}
4. gerar a Solução Inicial ($m^{(0)}$, $l^{(0)}$);
5.
6. {BT: vizinhanças}
7. $c_{bt} \leftarrow 0$;
8. **enquanto** $c_{bt} < 5$ **faça**
9. realizar Busca Tabu na lista de atividades;
10. realizar Busca Tabu na lista de modos;
11. $c_{bt} \leftarrow c_{bt} + 1$;
12. **fim enquanto**
13.
14. {Reconexão de Caminhos}
15. realizar a reconexão de caminhos a partir das soluções elite encontradas na BT;

Figura 5 – Estrutura da Implementação

As instâncias do problema fornecem os dados de entrada: um *upper bound* $T_{máx}$ para o tempo do projeto, o conjunto de atividades J, as precedências $\boldsymbol{P}_j$, as disponibilidades de recursos por tipo e qualidade (renovável ou não renovável), os modos possíveis para cada atividade e, para cada modo, a duração de cada atividade e as demandas de recursos por tipo e qualidade. As restrições de precedência, ou o conjunto de predecessoras diretas $\boldsymbol{P}_j$ das atividades $j \in J$, são geradas a partir dos conjuntos de atividades sucessoras diretas de cada atividade, formato este adotado para apresentação das precedências nas instâncias do problema.

Em todo o decorrer da implementação, ou seja, a partir da solução inicial, durante a Busca Tabu e na intensificação pela Reconexão de Caminhos, o cálculo do makespan de cada solução, representada pelo par de listas (m,l), é realizado pelo procedimento Construct (figura 5). Este módulo constrói o cronograma do projeto, a partir do esquema (m,l), estabelecendo os instantes de início e de fim de cada atividade, sempre verificando a viabilidade da solução, com relação ao tempo máximo $T_{máx}$ (linha 23) , às restrições de precedência (linha 8) e à disponibilidade de recursos renováveis no tempo (linha 21). Para o Construct, somente a restrição de tempo máximo $T_{máx}$ determina a inviabilidade da solução. Como será visto adiante, na descrição da Busca Tabu, quando ocorre a busca

da melhor solução em uma vizinhança, a solução corrente não é considerada na avaliação de melhoria. A comparação, para a escolha da melhor, ocorre somente entre as soluções obtidas na vizinhança da solução corrente. Desta forma, durante a evolução da Busca Tabu, podem ocorrer melhorias ou perdas de resultados em relação a cada solução corrente. Novamente, somente as soluções onde ocorre estouro do tempo máximo são desconsideradas.

No procedimento Construct foram estabelecidos controles para o uso de recursos renováveis em cada atividade escalonada, para que não fossem ultrapassados os limites R^{re}_k durante a construção do esquema solução; ocorrendo violação da restrição $\sum r^{re}_{k,mj} \leq R^{re}_k$ em algum instante de p_{mj} durante o escalonamento da atividade j, procurou-se escalonar a atividade no primeiro trecho de tempo completo disponível. Na medida em que ocorresse impossibilidade de escalonamento, por ausência de alguma atividade precedente entre as já escalonadas (linha 15), a atividade deve aguardar (fila de espera) o instante imediato ao escalonamento da última atividade precedente (linha 8).

Construct

1. Entrada: esquema (m,l), T_{max} ;
2. Saída: atividades escalonadas e o makespan total
3.
4. k ← 0;
5. **enquanto** k < |J| **faça**
6. **se** houver fila de espera // atividades aguardando precedentes
7. **então**
8. verificar se antecedentes foram escalonados;
9. i ← atividade a escalonar;
10. retirar da fila de espera;
11. **senão**
12. k ← k + 1;
13. i ← l(k) ; // atividade de ordem k da lista de atividades
14. **fim se**
15. **se** antecedentes não escalonados // falta antecedentes
16. **então**
17. colocar atividade na fila de espera
18. k ← k + 1;
19. **senão**
20. $t' \leftarrow \{máx(c_j) \mid$ **j** precede **i**, e foi escalonada no instante término de $c_j\}$
21. $s_i \leftarrow \{min(t) \mid t \geq t'$ e $r^{re}_{k,mj}$ é disponível nos p_{mi} instantes seguintes consecutivos$\}$
22. $c_i \leftarrow s_i + p_{mi}$;

23.	**se** $c_i > T_{max}$ **então** fim // excedeu restrição tempo T_{max}
24.	**fim se**
25.	**fim enquanto**
26.	retornar c_k ;

Figura 6 – Algoritmo do *Construct*

A Solução Inicial, composta por $m^{(0)}$ e $l^{(0)}$, respectivamente lista de modos e lista de atividades, é gerada em duas etapas (figura 6). A lista de modos $m^{(0)}$ é preenchida pelos modos m_j de cada atividade j, escolhidos com base na respectiva duração p_{mj} da atividade (linnha 11) e nas quantidades demandadas dos k tipos de recursos não renováveis $r^{non}_{k,mj}$ (linha 15). Esta escolha é feita de modo a que solução seja viável, e que propicie atividades com menor tempo e com menor uso de recursos não renováveis. A complexidade reside no fato de que estas duas últimas quantidades, de tempo e de recursos não renováveis, são conflitantes, ou seja, quanto maior o tempo, menor o uso de recursos, e vice-versa. O que dificulta a conciliação de menor tempo total (makespan) e uso de recursos (viabilidade).

Por sua vez, a lista de atividades $l^{(0)}$, que é uma lista ordenada das atividades, classificadas pela seqüência em que serão escalonadas, é preenchida (linha 22) utilizando-se a regra MTS (Most Total Successors), ou seja, as primeiras atividades na lista ordenada de atividades, que também são as primeiras a serem escalonadas, são as que apresentam o maior número de sucessores diretos ou indiretos. Na construção da solução inicial da BT, buscou-se uma solução que atendesse simultaneamente às restrições de tempo máximo, de precedência e de recursos não-renováveis, já que os renováveis só apresentam restrição na medida que o esquema da solução é construído.

Os principais blocos de algoritmos da implementação são a Busca Tabu da Solução Inicial, a Busca Tabu para explorar as vizinhanças e a Reconexão de Caminhos, detalhadas nas seções a seguir.

3.2.1. Descrição da Solução Inicial

Como a solução inicial é separada em duas estruturas $m^{(0)}$ e $l^{(0)}$ conforme descrito anteriormente, sobre a lista de modos $m^{(0)}$ é aplicado uma primeira Busca Tabu, pois a

construção de uma solução inicial viável com base na lista de modos também é um problema NP-Difícil [34].

BT: Solução Inicial ($m^{(0)}$, $l^{(0)}$)

1.
2. {Solução Inicial: lista modos}
3. $m^{(0)}$ = { $m_j \in M_j \mid \forall j \in J$, onde m_j equivale ao menor peso no uso de recursos};
4. {Busca Tabu sobre o vetor de modos}
5. $c_{si} \leftarrow 0$;
6. **enquanto** $c_{si} < 5$ **faça**
7. $k \leftarrow 0$;
8. **enquanto** $k < |J|$ **faça**
9. **para todo** $j \in J$ **faça**
10. **para todo** $m'_j \neq m_j \in M_j$ **faça**
11. escolher modo m'_{j*} não tabu tal que (p_{mj*} - $p_{m'j*}$) seja máximo;
12. **fim para**
13. **fim para**
14. lista tabu $\leftarrow (j^*, m'_{j*})$; // guardar atividade j^* e modo m'_{j*} na lista tabu;
15. **se (**solução é viável para todo $r^{non}_{k,mj*}$**) então**
16. $m_{j*} \leftarrow m'_{j*}$;
17. $k \leftarrow k + 1$;
18. **fim enquanto**
19. $c_{si} \leftarrow c_{si} + 1$;
20. **fim enquanto**
21. {Solução Inicial: lista de atividades}
22. $l^{(0)}$ = {l_{uj}= $j \in J$ | onde u_j = ordem da atividade j na lista l, quanto menor for u_j, maior a quantidade de sucessores diretos ou indiretos da atividade j}

Figura 7 – Algoritmo da Solução Inicial com a Busca Tabu

Para a aplicação de uma Busca Tabu especificamente sobre a lista de modos $m^{(0)}$, é estabelecida uma solução inicial para $m^{(0)}$: em cada atividade, o modo escolhido é aquele que utiliza o menor percentual de recursos não renováveis em relação ao total disponível (linha 3). A BT da solução inicial (figura 6) ocorreu apenas sobre a estrutura da lista de modos das atividades, com base em uma versão da Busca Tabu [37] aplicada através de uma abordagem genérica de um Problema de Satisfação de Restrições. A vizinhança foi percorrida em 5 ciclos (linha 6), aplicando-se o procedimento de troca de modo, que muda o modo m_j, associado à atividade j, para m'_j, de modo a obter melhorias mantendo-se a viabilidade, para que, em cada modo m_j possível, ocorra uma redução

equilibrada do tempo p_{mj} de cada atividade e da utilização de recursos não-renováveis $r^{non}_{k,mj}$, de cada tipo k no mesmo modo $m_{j.}$ Isto porque o tempo da atividade e a quantidade de recursos demandados são inversamente proporcionais. Já a utilização de recursos renováveis foi ignorada, pois restringe apenas o tempo máximo do projeto, já que depende da distribuição das atividades ao longo do tempo, no decorrer do cronograma do projeto, o que ainda não pode ser definido nesta primeira etapa da solução inicial.

Nesta busca tabu, percorrem-se as possibilidades de troca de modos de todas as atividades (linha 8), escolhe-se a não tabu que gera maior redução de tempo (linha 11), que é incluída na lista tabu (linha 14), verifica-se a viabilidade da solução (linha 15), executa-se a modificação e é dado prosseguimento na busca tabu.

3.2.2. Descrição da Busca Tabu para exploração da vizinhança (BT)

Um primeiro obstáculo encontrado no trabalho foi a definição da estruturação da implementação da BT, ou seja, como formatar a busca pela vizinhança, já que os artigos de Nonobe e Ibaraki só evidenciaram os operadores, mas não como ocorreriam as buscas. Optou-se pela aplicação alternada dos dois operadores, em blocos, sobre as respectivas estruturas de representação da solução: ChangeMode sobre a lista de modos e ShiftForward sobre a lista de atividades.

Ainda quanto à BT, em cada iteração, escolheu-se o melhor vizinho encontrado, em relação somente aos resultados alcançados na vizinhança, independentemente do makespan da solução corrente. Uma vez escolhida o melhor vizinho, ele passa a ser a solução corrente.

Após estabelecida a solução inicial, ocorre uma busca local em cinco ciclos, também para aumentar a profundidade de busca. Cada ciclo é composto por dois blocos consecutivos de varreduras: um primeiro bloco pesquisa a vizinhança da lista de atividades (figura 7) em 1000 iterações, com o uso do operador ShiftForward(A,B), que posiciona atividade B imediatamente antes da atividade A na lista de atividades; um segundo bloco pesquisa a vizinhança da lista de modos (figuras 8 e 9) em 1000 iterações, com o uso do operador ChangeMode(A,m'_A), que modifica o modo da atividade A, do corrente m_A para m'_A na lista de modos. As 1000 iterações também

procuram aprofundar a busca, e a execução consecutiva dos dois blocos procura inserir uma diversificação na busca.

A BT sobre a Lista de Atividades (figura 7) cobre, em cada uma das 1000 iterações (linha 2), uma quantidade |J| de vizinhanças da solução corrente (linha 4), ou seja, percorre todas as atividades da solução corrente, aleatoriamente, e escolhe uma posterior a ela, na ordem da lista de atividades, que não seja sucessora - direta ou indireta - (linhas 5 e 6) e que apresente a melhor redução de makespan quando aplicado o operador ShiftForward entre estas duas atividades (linha 8). Como foi explicado anteriormente, o processo Construct abandona soluções que ultrapassam o tempo máximo de processamento, sendo esta a única inviabilidade de uma solução durante a Busca Tabu (linha 9). Por outro lado, as soluções com violação das restrições de recursos são mantidas no elenco a ser considerado pelo procedimento. Assim, para cada atividade A escolhida aleatoriamente, se reúne o conjunto A_{pos}, com as atividades posteriores a A na ordem da lista de atividades, considerando-se aquelas que não são sucessoras diretas ou indiretas de A (pelas restrições de precedência) e cujo par de atividades – A escolhida e $B \in A_{pos}$ – não pertence à lista tabu. Para cada atividade de A_{pos}, verifica, isoladamente, a troca de ordem e escolhe a atividade B que apresentar solução viável com o menor makespan para a troca correspondente (linha 12). Após escolher |J| pares (A,B), e, conseqüentemente, tendo avaliado todas as modificações de ordem possíveis da vizinhança da solução corrente, opta por aquela que gera o menor makespan (linha 16) e avança, ou seja, efetiva a troca na lista de atividades (linha 17), inclui o par de atividades (A,B) na lista tabu (linha 18), e prossegue com a busca na próxima vizinhança, até completar 1000 iterações.

BT: lista de atividades

1. $c_{la} \leftarrow 0$;
2. **enquanto** $c_{la} < 1000$ **faça**
3. $k \leftarrow 0$;
4. **enquanto** $k < |J|$ **faça**
5. escolher aleatoriamente atividade $A \in J$, de posição u_A na lista de atividades;
6. $A_{pos} \leftarrow \{i \in J \mid u_i > u_A$ na lista de atividades, i não é sucessor direto ou indireto de A, e par (A,i) não tabu$\}$
7. **para todo** $i \in A_{pos}$ **faça**
8. $l_{Ai} \leftarrow$ ShiftForward(A,i); //{reposiciona i imediatamente antes de A na lista de atividades l }
9. $makespan_{Ai} \leftarrow$ Construct (m,l_{Ai}); //{se solução for viável}
10. volta posições originais na lista de atividades; // desfaz ShiftForward(A,i)
11. **fim para**
12. $B \leftarrow i$; //{$i \in A_{pos}$ | corresponde ao mínimo($makespan_{Ai}$)}
13. $makespan_{AB} \leftarrow$ mínimo($makespan_{Ai}$);
14. $k \leftarrow k + 1$;
15. **fim enquanto**
16. selecionar par de atividades A e B que apresentaram menor $makespan_{AB}$;
17. l ← ShiftForward(A,B); //{reposiciona B imediatamente antes de A na lista de atividades l}
18. lista tabu ← (A,B); // acrescenta par (A,B) na lista tabu
19. $c_{la} \leftarrow c_{la} + 1$;
20. **fim enquanto**

Figura 8 – Algoritmo da Busca Tabu na Lista de Atividades

Após inúmeras avaliações dos resultados da BT sobre a Lista de Atividades, foi verificado que a troca de ordem das atividades (ShiftForward) propiciava pouca ou, muitas vezes, nenhuma melhoria nas soluções. Assim, optou-se por gerar duas versões diferentes da implementação: uma ativando a BT sobre a lista de atividades, com o operador ShiftForward, e outra não.

Assim como na BT sobre a Lista de Atividades, também na BT sobre a Lista de Modos foram implementadas duas versões, cobrindo-se a vizinhança por dois critérios distintos: em uma, avalia-se primeiro todas as trocas de modos possíveis da atividade escolhida aleatoriamente e, em seguida, avança-se para o modo que acarretar o menor makespan, antes mesmo de avaliar os modos das demais atividades da solução corrente (figura 8), ou seja, o avanço ocorre a cada troca de modo que acarrete redução de tempo da

atividade que foi selecionada aleatoriamente; em outra, escolhe-se aleatoriamente a ordem e, varrendo-se toda a vizinhança, troca-se o modo das atividades da vizinhança para aquele que gerar o menor makespan (figura 9), ou seja, o avanço só ocorre após a verificação de todos os modos das atividades da solução corrente.

A primeira versão, mostrada na figura 8, cobre, em cada uma das 1000 iterações (linha 2), uma quantidade |J| de vizinhos da solução corrente (linha 4), ou seja, percorre todas as atividades da solução corrente, aleatoriamente, e avalia todas as trocas de modos possíveis para cada atividade utilizando o operador ChangeMode(A,m'_A), trocando o modo da atividade A de m_A para m'_A (linha 7). Para cada atividade visitada, escolhe o modo que, quando aplicada a troca, apresentou menor makespan (linha 11), e avança para este modo aplicando a troca na lista de modos m (linha 12) e armazenando o par (A,m'_A) na lista tabu (linha 13). Prossegue, assim, em todas as |J| atividades da solução corrente. Repete esta busca por 1000 iterações. Esta versão da heurística é gulosa, ou seja, avança para o primeiro ganho de tempo (makespan) que encontra, executando a troca correspondente na lista de modos.

Já na segunda versão da BT (figura 9), o algoritmo só avança na vizinhança, ou seja, a troca de modo só é efetivada (linha 13) ao final das |J| atividades visitadas, selecionando aquela que, quando aplicada a troca de modos pelo operador ChangeMode(A,m'A), apresente a melhor redução de makespan (linha 12).

Dada a característica dos avanços proporcionados pelos operadores, tanto nas listas de modos, quanto, principalmente, nas listas de atividades, que geravam poucos avanços de soluções viáveis, estabeleceram-se dois modelos para a aplicação das Listas Tabus: um primeiro, que comportou, numa lista dinâmica, todos os movimentos realizados, sem implementar critério de aspiração ou tamanho máximo para a Lista Tabu; um segundo, onde o Prazo Tabu foi implementado considerando-se dois tamanhos de ciclos distintos (uma versão com Prazo Tabu de 30 ciclos, e outra de 1500 ciclos). Portanto, foram implementadas 3 versões para a BT na lista de modos. Para melhor comparar a performance entre elas, foram implementadas somente nas versões em que a BT da lista de atividades não foi ativada, conforme será visto na seção 3.2.4.

BT: lista de modos

1. $c_{lm} \leftarrow 0$;
2. **enquanto** $c_{lm} < 1000$ **faça**
3. $k \leftarrow 0$;
4. **enquanto** $k < |J|$ **faça**
5. escolher aleatoriamente atividade $A \in J$;
6. **para todo** $m'_A \neq m_A \in M_A$ **faça**
7. $m_{m'A} \leftarrow$ ChangeMode(A,m'_A); //{troca modo da atividade A para m'_A}
8. $makespan_{m'A} \leftarrow$ Construct ($m_{m'A}$,l); //{se solução for viável}
9. volta modo original na lista de modos; //desfaz ChangeMode(A,m'_A)
10. **fim para**
11. selecionar atividade A e modo m'_A que apresentaram menor $makespan_{m'A}$;
12. m $\leftarrow$ ChangeMode(A,m'_A); //{ troca modo da atividade A para m'_A }
13. lista tabu $\leftarrow$ (A,m'_A); // acrescenta par (A, m'_A) na lista tabu
14. **fim enquanto**
15. $c_{lm} \leftarrow c_{lm} + 1$;
16. **fim enquanto**

Figura 9 – Algoritmo da Busca Tabu na Lista de Modos – heurística gulosa

BT: lista de modos

1. $c_{lm} \leftarrow 0$;
2. **enquanto** $c_{lm} < 1000$ **faça**
3. $k \leftarrow 0$;
4. **enquanto** $k < |J|$ **faça**
5. escolher aleatoriamente atividade $A \in J$;
6. **para todo** $m'_A \neq m_A \in M_A$ **faça**
7. $m_{m'A} \leftarrow$ ChangeMode(A,m'_A); //{troca modo da atividade A para m'_A}
8. $makespan_{m'A} \leftarrow$ Construct ($m_{m'A}$,l); //{se solução for viável}
9. volta modo original na lista de modos; //desfaz ChangeMode(A,m'_A)
10. **fim para**
11. **fim enquanto**
12. selecionar atividade A e modo m'_A que apresentaram menor $makespan_{m'A}$;
13. m $\leftarrow$ ChangeMode(A,m'_A); //{ troca modo da atividade A para m'_A }
14. lista tabu $\leftarrow$ (A,m'_A); // acrescenta par (A, m'_A) na lista tabu
15. $c_{lm} \leftarrow c_{lm} + 1$;
16. **fim enquanto**

Figura 10 – Algoritmo da Busca Tabu na Lista de Modos – heurística não gulosa

3.2.3. Descrição da Reconexão de Caminhos (RC)

A Reconexão de Caminhos foi construída em uma versão pós-otimização, e parte de todas as soluções elite obtidas no decorrer do Tabu. Optou-se, também, por um caminho de verificação somente sobre pares de soluções elite consecutivas, partindo da pior solução (inicial) para a melhor (guia). Com a redução das buscas sobre os caminhos ocorrendo somente entre pares de soluções consecutivas (valores próximos), buscou-se uma economia no tempo de processamento, pela restrição do número de soluções intermediárias a serem avaliadas [40][30].

Reconexão de Caminhos

1. Entrada: vetores com as listas **l** e **m** das soluções elite encontradas no decorrer do Tabu. O vetor é ordenado inversamente ao makespan resultante da solução elite, ou seja, do pior para o melhor.
2.
3. $k \leftarrow 1$, $c_{best} \leftarrow$ quantidade de soluções elite;
4. best ← ultima solução elite do tabu
5. **enquanto** $k < c_{best}$ **faça**
6. RC_inicial $\leftarrow m_K$; // lista de modos da solução elite de ordem k
7. $l_{RC} \leftarrow l_k$; // lista de atividades da solução elite de ordem k
8. RC ← RC_inicial ; // lista de modos da solução elite de ordem k
9. RC_guia $\leftarrow m_{K+1}$; // lista de modos da solução elite de ordem k+1
10. **enquanto** RC ≠ RC_guia **faça**
11. RC′ ← RC;
12. **para todo** m'_j em RC′ ≠ m_j em RC_guia , onde $j \in J$ **faça**
13. RC′ ← ChangeMode(j,m_j); //
14. $makespan_{RCj}$ ← Construct (RC′, l_{RC}); //{se solução for viável}
15. **fim para**
16. selecionar atividade j e modo m_j que apresentaram menor $makespan_{RCj}$;
17. RC ← ChangeMode(j,m_j) em RC;
18. se $makespan_{RCj} <$ best então
19. best ← $makespan_{RCj}$;
20. $best_{modos} \leftarrow$ RC;
21. $best_{atividades} \leftarrow l_{RC}$;
22. **fim enquanto**
23. **fim enquanto**

Figura 11 – Algoritmo da Reconexão de Caminhos

Ao final da fase de Busca Tabu, as soluções elite estão reunidas em um vetor, estando a pior (solução inicial) na primeira posição do vetor, e a melhor (última obtida pela BT) ao final. Para cada solução elite são armazenados o makespan correspondente, a lista de atividades l e a lista de modos m. Desta forma, os valores de *makespan* correspondentes estarão em ordem decrescente. Para cada par de soluções consecutivas neste vetor, é realizada a Reconexão de Caminhos.

Como a troca de ordem sobre a lista de atividade gerou poucos resultados significativos na vizinhança da BT, buscou-se aplicar apenas troca de modos na lista de modos para estabelecer o caminho a ser percorrido entre as soluções elite. Assim, apesar de percorrermos o caminho até quase igualarmos as listas de modos entre as soluções intermediária e guia da RC, as respectivas listas de atividade serão (ou poderão ser) distintas.

Para cada par de soluções elite consecutivas, a RC estabelece uma solução intermediária executando trocas consecutivas de modos de atividades (linha 10) que sejam distintos entre a solução inicial e a solução guia, dando prioridade àquelas trocas que acarretem maior redução de makespan (linha 16). A cada troca, avalia também a viabilidade de recursos disponíveis e de restrição de tempo na solução correspondente (através do procedimento Construct). As trocas de modos são incorporadas, sucessivamente, à lista de modos da solução intermediária da RC (linha 17). É importante lembrar que a lista de atividades correspondente à solução intermediária da RC é a lista da solução inicial do par de elites. Sempre que uma solução intermediária apresentar makespan inferior aos já apurados pela RC e ao conjunto de soluções elite da BT (linha 18), é definida como a melhor solução (*best*).

3.2.4. Implementação

No decorrer do trabalho, foram definidas algumas opções e critérios, de acordo com os objetivos a serem alcançados, conforme mostrado nas seções anteriores deste capítulo.

Para avaliar a correção das soluções, paralelamente à implementação, foi criada uma construção visual automática do esquema solução, com controles das quantidades disponíveis e utilizadas de recursos R^{re}_k no decorrer do projeto, tendo como entrada os dados das instancias e os resultados gerados pelo algoritmo Construct para cada

atividade j, ou seja, s_j (instante de início), p_{mj} (duração), m_j (modo de execução), $r^{re}_{k,mj}$ (quantidades demandadas de recursos renováveis do tipo k) e $r^{non}_{k,mj}$ (quantidades demandadas de recursos não renováveis do tipo k).

Como mencionado na seção 3.2.1, as buscas da vizinhança (Change-Mode/Lista de Modos e ShiftForward/Lista de Atividades) apresentaram perfis bastante diferenciados na atuação sobre os resultados. As trocas de modos sempre alteram significativamente as quantidades de recursos utilizados e também o makespan total do projeto; assim, a dificuldade reside em equilibrar a redução de tempo com a ampliação da utilização de R^{non}, e quase sempre evoluíam tendendo a piorar bastante a solução, tornando-a inviável, sem retorno a um ponto de viabilidade. Por sua vez, a troca de ordem das atividades não alterava o consumo de recursos não renováveis, raramente os renováveis e, quase nunca, o tempo do projeto. Entretanto, gerava variações bastante significativas na utilização dos recursos disponíveis, tanto renováveis quanto não renováveis, propiciando surgimento de soluções inviáveis. Sobre as trocas de modos, ainda conforme a seção 3.2.1, foram elaborados dois critérios para a evolução na vizinhança: uma varrendo toda a vizinhança da solução corrente antes de evoluir, e outro, guloso, evoluindo na primeira viabilidade da vizinhança da solução corrente. Adicionalmente, para cada um destes dois critérios, foram consideradas 3 implementações para a Lista Tabu: uma primeira com Prazo Tabu infinito, permanecendo os movimentos indefinidamente na Lista Tabu (dinâmica), uma segunda com Prazo Tabu de 30 ciclos e, por fim, uma terceira com Prazo Tabu de 1500 ciclos. Deste modo, foram geradas e processadas oito versões da implementação. A troca de ordem das atividades (figura 7) foi implementada nas versões V3 e V4, ficando ausente nas versões V1, V1.a, V1.b, V2, V2.a e V2.b. Por sua vez, a troca de modos foi implementada diferentemente nas versões V4 e grupo V1 (figura 8) das versões V3 e grupo V2 (figura 9), conforme descrito na seção anterior e indicado na tabela 4. As quatro implementações forneceram, para cada instância: as soluções elite, incluindo a solução inicial, no formato (m,l); os respectivos instantes de início e término de cada atividade escalonada pela solução; os respectivos *makespans* alcançados; os horários de início e de término do processamento total e dos processamentos dos principais blocos (Solução Inicial, Busca Tabu vizinhança, Reconexão de Caminhos), como também o horário em que cada solução elite foi obtida.

versões	troca de ordem	troca de modos	prazo tabu
V1	não	gulosa	-
V1.a	não	gulosa	30
V1.b	não	gulosa	1.500
V2	não	não gulosa	-
V2.a	não	não gulosa	30
V2.b	não	não gulosa	1.500
V3	sim	não gulosa	-
V4	sim	gulosa	-

Tabela 4 – Características das oito versões da implementação

Nesta combinação de versões dos algoritmos de BT (Tabela 4), a versão V1 da implementação foi a que apresentou o melhor resultado, conforme apresentado na seção a seguir.

4. Resultados Computacionais

Abordando o problema do Problema de Escalonamento de Atividades de Projetos sob Restrição de Recursos, na versão multimodo, onde as atividades podem ser executadas por diferentes modalidades, com características de duração e de restrições distintas entre si, o principal objetivo do trabalho foi o de obter boas soluções através do uso da Busca Tabu intensificada pelo uso da Reconexão de Caminhos.

A aplicação foi avaliada processando-se instâncias do problema, disponíveis em PSPLIB (http://129.187.106.231/psplib/). Nesta biblioteca são encontradas diversas instâncias agrupadas por quantidade de atividades do projeto, por tipo de geração (distinção entre parâmetros de entrada para a geração das instâncias), por complexidade do problema (simples, multimodo, com *time lags*, etc.), além dos resultados (menor makespan) já encontrados para cada uma delas, podendo ser soluções ótimas, soluções heurísticas ou limites mínimos (*lower bounds*), sendo indicado também o autor e o método utilizado. Para este trabalho foi considerado somente o caso multimodo, especificamente o grupo j30mm, com instâncias de 30 atividades. Algumas instâncias, utilizadas em trabalhos anteriores, não foram contempladas por fugirem ao escopo deste trabalho: grupos de modo simples (único modo) j20sm, j60sm e j90sm, respectivamente com 20, 60 e 90 atividades, e grupo mm100, com 100 atividades multimodo, porém com restrição de tempo máximo entre atividades precedentes.

As instancias de 30 atividades são compostas por quantidades limitadas, e distintas entre elas, de recursos renováveis e não renováveis, ambos de 2 tipos cada. Além disso, existem 3 modos para cada atividade e, no máximo, 3 atividades sucessoras (restrições de precedência). Cada modo possui três grupos de informações: duração da atividade, demandas de recursos renováveis e demandas de recursos não renováveis, sendo estes dois últimos separados por 2 tipos cada. Como restrição de tempo, apresentam um tempo máximo (*upper bound*) limite para a execução do projeto. São 64 grupos, compostos de 10 instâncias cada, estas geradas por parâmetros similares dentro de cada grupo. Desta forma, totalizam 640 instâncias, para as quais existem, no benchmark, 552 resultados, obtidos por diversos autores. As demais 88 instâncias permanecem sem resultados viáveis, que cumpram as restrições de tempo – *upper bound* – e de recursos não renováveis.

benchmark		versões implementadas							
instancia	best	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
301_1 a 307_7	-	-	-	-	-	-	-	-	-
307_8	**47**	57	57	57	57	57	57	57	**50**
307_9	-	-	-	-	-	-	-	-	-
307_10	-	-	-	-	-	-	-	-	-
308_1	-	-	-	-	-	-	-	-	-
308_2	-	-	-	-	-	-	-	-	-
308_3	-	-	-	-	-	-	-	-	-
308_4	-	-	-	-	-	-	-	-	-
308_5	-	-	-	-	-	-	-	-	-
308_6	**46**	**46**	**46**	**46**	48	48	48	48	**46**
308_7	-	-	-	-	-	-	-	-	-
308_8	-	-	-	-	-	-	-	-	-
308_9	-	-	-	-	-	-	-	-	-
308_10	-	-	-	-	-	-	-	-	-
309_1	**31**	**33**	**33**	39	39	39	39	36	34
309_2	**27**	32	32	32	32	32	32	**30**	32
309_3	**34**	**35**	**35**	36	36	36	36	39	36
309_4	**30**	33	33	34	34	34	34	38	**32**
309_5	**27**	31	32	32	31	31	31	**30**	32
309_6	**31**	**34**	**34**	35	37	37	37	36	**34**
309_7	**35**	**35**	**35**	41	38	38	38	38	36
309_8	**26**	**28**	**28**	32	31	31	31	31	31
309_9	**32**	-	-	-	-	-	-	-	-
309_10	**25**	**27**	29	30	30	30	30	30	28
3010_1	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**26**
3010_2	**28**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3010_3	**24**	26	26	26	25	25	25	**24**	25
3010_4	**36**	**36**	37	37	**36**	**36**	**36**	**36**	**36**
3010_5	**33**	**34**	35	37	37	37	37	37	**34**
3010_6	**25**	**25**	**25**	27	28	28	28	28	**25**
3010_7	**22**	23	23	23	**22**	**22**	**22**	**22**	**22**
3010_8	**31**	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**
3010_9	**38**	44	42	42	**40**	**40**	**40**	**40**	**40**
3010_10	**32**	**33**	**33**	38	37	37	37	37	**33**
3011_1	**35**	**35**	**35**	**35**	40	40	40	40	**35**
3011_2	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
3011_3	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
3011_4	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**
3011_5	**35**	**35**	**35**	**35**	38	38	38	38	**35**
3011_6	**33**	**33**	36	36	38	38	38	38	**33**
3011_7	**32**	33	**32**	**32**	36	36	36	36	**32**
3011_8	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
3011_9	**26**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3011_10	**35**	39	39	39	39	39	39	39	**35**
3012_1	**29**	31	31	31	**29**	**29**	**29**	**29**	31
3012_2	**31**	**31**	**31**	36	**31**	**31**	**31**	**31**	**31**
3012_3	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**
3012_4	**29**	**29**	30	30	32	32	32	32	31
3012_5	**25**	27	27	27	**26**	**26**	**26**	**26**	27
3012_6	**34**	**34**	**34**	41	41	41	41	41	**34**
3012_7	**27**	30	31	**28**	31	31	31	31	29
3012_8	**28**	**28**	**28**	29	29	29	29	29	29
3012_9	**22**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
3012_10	**28**	**28**	**28**	**28**	29	29	29	29	**28**
3013_1	**37**	41	44	44	42	42	42	**40**	**40**
3013_2	**40**	**44**	46	49	46	46	46	47	45
3013_3	**41**	46	45	48	47	47	47	**44**	**44**
3013_4	**42**	46	47	52	48	48	48	**44**	45
3013_5	**38**	42	45	45	42	42	42	**39**	41
3013_6	**38**	43	44	52	44	44	44	**41**	44
3013_7	**41**	47	47	50	48	48	48	**44**	**44**
3013_8	**28**	30	30	30	29	29	29	**28**	**28**
3013_9	**30**	34	34	34	34	34	34	**31**	33
3013_10	**43**	**47**	**47**	51	54	54	54	50	48
3014_1	**37**	40	40	40	39	39	39	**37**	**37**
3014_2	**32**	**33**	34	35	34	34	34	34	34
3014_3	**30**	**32**	**32**	35	34	34	34	33	33
3014_4	**34**	**34**	35	39	39	39	39	38	36
3014_5	**26**	**29**	**29**	32	32	32	32	31	**29**
3014_6	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3014_7	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**
3014_8	**41**	**42**	**42**	43	43	43	43	43	**42**
3014_9	**31**	35	34	34	34	34	34	**32**	**32**
3014_10	**29**	32	31	34	33	33	33	**29**	30
3015_1	**40**	**40**	**40**	**40**	42	42	42	42	**40**
3015_2	**34**	36	35	36	36	36	36	36	**34**
3015_3	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**
3015_4	**30**	**30**	**30**	33	40	40	40	40	33
3015_5	**24**	**26**	27	30	28	28	28	27	27
3015_6	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**
3015_7	**27**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3015_8	**23**	**23**	**23**	25	25	25	25	25	**23**
3015_9	**28**	30	31	31	30	30	30	**29**	31
3015_10	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**38**
3016_1	**27**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
3016_2	**41**	**41**	**41**	**41**	42	42	42	42	**41**
3016_3	**23**	26	26	26	29	29	29	29	**25**
3016_4	**19**	21	21	21	20	20	20	**19**	20
3016_5	**30**	**31**	33	33	33	33	33	33	33
3016_6	**30**	**30**	**30**	**30**	34	34	34	34	34
3016_7	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**
3016_8	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
3016_9	**24**	-	-	-	-	-	-	-	-
3016_10	**32**	**32**	**32**	33	33	33	33	33	**32**

benchmark		versões implementadas							
instancia	best	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
3017_1	**34**	**35**	**35**	41	36	36	36	36	37
3017_2	**26**	27	**26**	30	27	27	27	27	27
3017_3	**33**	**33**	34	36	36	36	36	34	34
3017_4	**39**	**40**	**40**	43	41	41	41	41	41
3017_5	**29**	**29**	**29**	33	**29**	**29**	**29**	**29**	**29**
3017_6	**33**	**36**	**36**	39	**36**	**36**	**36**	**36**	**36**
3017_7	**32**	**32**	33	36	33	33	33	33	33
3017_8	**31**	34	**32**	34	34	34	34	**32**	**32**
3017_9	**34**	**35**	**35**	36	42	42	42	39	**35**
3017_10	**26**	**28**	**28**	30	30	30	30	30	**28**
3018_1	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
3018_2	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**
3018_3	**35**	**35**	**35**	36	36	36	36	36	**35**
3018_4	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
3018_5	**21**	24	24	24	23	23	23	**21**	24
3018_6	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
3018_7	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**
3018_8	**18**	20	**18**	21	**18**	**18**	**18**	**18**	**18**
3018_9	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	27
3018_10	**33**	**33**	**33**	37	36	36	36	**33**	34
3019_1	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**
3019_2	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**
3019_3	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
3019_4	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
3019_5	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**
3019_6	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**
3019_7	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**
3019_8	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**
3019_9	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
3019_10	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3020_1	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
3020_2	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
3020_3	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**
3020_4	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**
3020_5	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3020_6	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**
3020_7	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
3020_8	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**
3020_9	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**26**
3020_10	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
3021_1	**38**	44	43	46	45	45	45	**42**	**42**
3021_2	**34**	38	38	38	38	38	38	**36**	**36**
3021_3	**36**	41	43	43	41	41	41	**39**	**39**
3021_4	**37**	40	**39**	45	43	43	43	43	43
3021_5	**37**	41	41	44	**40**	**40**	**40**	**40**	**40**
3021_6	**33**	38	39	39	38	38	38	**37**	**37**
3021_7	**36**	41	41	41	39	39	39	**37**	**37**
3021_8	**44**	49	49	53	50	50	50	**46**	**46**
3021_9	**32**	36	37	37	37	37	37	**34**	**34**
3021_10	**45**	**49**	50	51	50	50	50	**49**	50
3022_1	**26**	29	28	28	27	27	27	**26**	**26**
3022_2	**36**	38	38	38	38	38	38	**37**	**37**
3022_3	**25**	26	26	29	26	26	26	**25**	**25**
3022_4	**26**	28	27	28	28	28	28	**26**	**26**
3022_5	**30**	33	33	33	33	33	33	**31**	**31**
3022_6	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
3022_7	**38**	**38**	39	39	41	41	41	39	39
3022_8	**33**	38	37	40	38	38	38	**35**	36
3022_9	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**
3022_10	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
3023_1	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
3023_2	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**
3023_3	**23**	24	24	24	24	24	24	**23**	**23**
3023_4	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**
3023_5	**32**	**32**	**32**	35	**32**	**32**	**32**	**32**	**32**
3023_6	**27**	28	28	28	**27**	**27**	**27**	**27**	**27**
3023_7	**24**	26	26	26	25	25	25	**24**	**24**
3023_8	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
3023_9	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**26**
3023_10	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**
3024_1	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**
3024_2	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
3024_3	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**
3024_4	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3024_5	**37**	**37**	**37**	**37**	**37**	**37**	**37**	**37**	**37**
3024_6	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
3024_7	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3024_8	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**38**
3024_9	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
3024_10	**37**	**37**	**37**	**37**	**37**	**37**	**37**	**37**	**37**
3025_1	**34**	**34**	37	38	**34**	**34**	**34**	**34**	36
3025_2	**32**	36	39	39	37	37	37	**35**	37
3025_3	**24**	27	**25**	**25**	**25**	**25**	**25**	**25**	26
3025_4	**30**	**30**	**30**	31	31	31	31	33	33
3025_5	**27**	31	**30**	35	34	34	34	32	31
3025_6	**32**	**32**	**32**	**32**	**32**	**32**	**32**	**32**	**32**
3025_7	**33**	**33**	**33**	36	**33**	**33**	**33**	**33**	**33**
3025_8	**33**	38	38	38	37	37	37	**33**	35
3025_9	**51**	**51**	**51**	53	**51**	**51**	**51**	**51**	**51**
3025_10	**27**	**27**	30	32	**27**	**27**	**27**	**27**	31
3026_1	**21**	24	**22**	24	**22**	**22**	**22**	**22**	**22**
3026_2	**22**	23	**22**	23	**22**	**22**	**22**	**22**	**22**
3026_3	**42**	**42**	**42**	**42**	**42**	**42**	**42**	**42**	**42**
3026_4	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**
3026_5	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**
3026_6	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**

benchmark		versões implementadas							
instancia	best	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
3026_7	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3026_8	**40**	**40**	**40**	**40**	**40**	**40**	**40**	**40**	**40**
3026_9	**21**	**22**	**22**	24	23	23	23	**22**	23
3026_10	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**
3027_1	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**38**
3027_2	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3027_3	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
3027_4	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
3027_5	**40**	**40**	**40**	**40**	**40**	**40**	**40**	**40**	**40**
3027_6	**25**	26	26	26	**25**	**25**	**25**	**25**	**25**
3027_7	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
3027_8	**42**	**42**	**42**	**42**	**42**	**42**	**42**	**42**	**42**
3027_9	**37**	**37**	**37**	**37**	**37**	**37**	**37**	**37**	**37**
3027_10	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3028_1	**37**	**37**	**37**	**37**	**37**	**37**	**37**	**37**	**37**
3028_2	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
3028_3	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
3028_4	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**
3028_5	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
3028_6	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3028_7	**37**	**37**	**37**	**37**	**37**	**37**	**37**	**37**	**37**
3028_8	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**
3028_9	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
3028_10	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**26**
3029_1	**46**	49	**48**	53	49	49	49	49	49
3029_2	**35**	39	**38**	42	39	39	39	**38**	**38**
3029_3	**32**	**33**	**33**	41	35	35	35	34	36
3029_4	**33**	36	37	42	**35**	**35**	**35**	39	39
3029_5	**34**	42	39	42	**36**	**36**	**36**	**36**	38
3029_6	**42**	**45**	46	46	46	46	46	**45**	49
3029_7	**38**	**41**	42	45	43	43	43	42	42
3029_8	**50**	57	58	58	56	56	56	**52**	54
3029_9	**37**	41	42	42	41	41	41	**38**	39
3029_10	**36**	**39**	**39**	44	**39**	**39**	**39**	40	40
3030_1	**36**	**36**	39	39	**36**	**36**	**36**	**36**	**36**
3030_2	**30**	33	32	32	32	32	32	**30**	31
3030_3	**29**	31	31	31	31	31	31	**30**	**30**
3030_4	**35**	**35**	36	36	**35**	**35**	**35**	**35**	**35**
3030_5	**30**	31	31	34	31	31	31	**30**	31
3030_6	**41**	44	44	44	**41**	**41**	**41**	**41**	**41**
3030_7	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**
3030_8	**28**	32	32	32	31	31	31	**29**	**29**
3030_9	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**
3030_10	**19**	20	20	20	20	20	20	**19**	**19**
3031_1	**25**	28	28	28	27	27	27	**26**	**26**
3031_2	**43**	**43**	**43**	**43**	**43**	**43**	**43**	**43**	**43**
3031_3	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
3031_4	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3031_5	**43**	**43**	**43**	**43**	**43**	**43**	**43**	**43**	**43**
3031_6	**32**	**32**	**32**	**32**	**32**	**32**	**32**	**32**	**32**
3031_7	**45**	**45**	**45**	**45**	**45**	**45**	**45**	**45**	**45**
3031_8	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**26**
3031_9	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**
3031_10	**20**	21	23	23	21	21	21	**20**	**20**
3032_1	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
3032_2	**32**	**32**	**32**	**32**	**32**	**32**	**32**	**32**	**32**
3032_3	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
3032_4	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**
3032_5	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3032_6	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3032_7	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**
3032_8	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3032_9	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**
3032_10	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3033_1	**47**	55	54	**53**	57	57	57	65	61
3033_2	**45**	61	52	52	57	57	57	57	**51**
3033_3	**45**	51	48	55	60	60	60	61	**46**
3033_4	**57**	70	66	67	72	72	72	72	**65**
3033_5	**48**	52	52	54	73	73	73	72	**51**
3033_6	**41**	47	**46**	53	54	54	54	-	-
3033_7	**45**	-	-	-	-	-	-	-	-
3033_8	**39**	45	46	46	52	52	52	46	**42**
3033_9	**44**	-	-	-	-	-	-	-	-
3033_10	**39**	**46**	**46**	49	49	49	49	51	**46**
3034_1	**39**	**43**	44	**43**	55	55	55	55	47
3034_2	**32**	39	36	41	44	44	44	44	**34**
3034_3	**38**	43	**41**	46	48	48	48	50	45
3034_4	**42**	-	-	-	-	-	-	-	-
3034_5	**43**	59	56	**54**	63	63	63	63	**54**
3034_6	**41**	**43**	44	46	51	51	51	51	46
3034_7	**44**	50	50	**48**	57	57	57	57	51
3034_8	**47**	-	-	-	-	-	-	-	-
3034_9	**38**	-	-	-	-	-	-	-	-
3034_10	**54**	-	-	-	-	-	-	-	-
3035_1	**34**	38	38	37	54	54	54	54	**36**
3035_2	**35**	-	-	-	-	-	-	-	-
3035_3	**39**	**43**	**43**	44	51	51	51	51	**43**
3035_4	**47**	53	**51**	53	59	59	59	59	**51**
3035_5	**42**	53	55	**52**	56	56	56	56	**52**
3035_6	**41**	-	-	-	-	-	-	-	-
3035_7	**35**	38	38	39	43	43	43	43	**37**
3035_8	**40**	49	47	49	55	55	55	51	**44**
3035_9	**36**	-	-	-	-	-	-	-	-
3035_10	**51**	63	65	62	72	72	72	72	**61**
3036_1	-	-	-	-	-	-	-	-	-
3036_2	-	-	-	-	-	-	-	-	-

Tabela 5: Resultados das oito versões da implementação vs benchmark

benchmark		versões implementadas							
instancia	best	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
3036_3	-	-	-	-	-	-	-	-	-
3036_4	-	-	-	-	-	-	-	-	-
3036_5	-	-	-	-	-	-	-	-	-
3036_6	-	-	-	-	-	-	-	-	-
3036_7	-	-	-	-	-	-	-	-	-
3036_8	-	-	-	-	-	-	-	-	-
3036_9	-	-	-	-	-	-	-	-	-
3036_10	-	-	-	-	-	-	-	-	-
3037_1	**54**	**60**	61	62	65	65	65	65	62
3037_2	**58**	68	68	68	71	71	71	71	**63**
3037_3	**56**	**62**	**62**	65	68	68	68	67	64
3037_4	**54**	63	63	68	66	66	66	62	**60**
3037_5	**51**	-	-	-	-	-	-	-	-
3037_6	**62**	67	68	67	67	67	67	**65**	**65**
3037_7	**57**	-	-	-	-	-	-	-	-
3037_8	**44**	51	**49**	50	58	58	58	53	**49**
3037_9	**59**	-	-	-	-	-	-	-	-
3037_10	**42**	**45**	**45**	49	49	49	49	48	47
3038_1	**46**	**50**	57	51	58	58	58	52	56
3038_2	**44**	55	49	49	51	51	51	51	**48**
3038_3	**34**	**41**	**41**	**41**	**41**	**41**	**41**	**41**	**41**
3038_4	**50**	55	**54**	56	63	63	63	56	55
3038_5	**50**	54	**53**	**53**	64	64	64	62	**53**
3038_6	**37**	-	-	-	-	-	-	-	-
3038_7	**38**	41	42	42	42	42	42	**39**	42
3038_8	**40**	44	44	**43**	58	58	58	58	44
3038_9	**45**	**51**	55	55	56	56	56	57	**51**
3038_10	**37**	**40**	**40**	43	47	47	47	45	42
3039_1	**43**	50	49	49	**48**	**48**	**48**	**48**	50
3039_2	**42**	48	50	48	51	51	51	51	**47**
3039_3	**49**	58	60	**57**	64	64	64	64	58
3039_4	**49**	**56**	58	57	57	57	57	57	**56**
3039_5	**32**	**35**	**35**	42	38	38	38	36	37
3039_6	**36**	-	-	-	-	-	-	-	-
3039_7	**38**	47	46	51	56	56	56	56	**42**
3039_8	**42**	50	**48**	**48**	50	50	50	50	**48**
3039_9	**42**	**47**	52	51	53	53	53	51	48
3039_10	**44**	**45**	52	52	60	60	60	48	51
3040_1	**38**	55	**43**	49	51	51	51	50	**43**
3040_2	**41**	**44**	**44**	45	47	47	47	47	**44**
3040_3	**36**	**39**	41	42	44	44	44	44	**39**
3040_4	**40**	47	44	44	53	53	53	53	**43**
3040_5	**48**	52	53	**51**	59	59	59	59	53
3040_6	**41**	-	-	-	-	-	-	-	-
3040_7	**48**	60	58	57	64	64	64	64	**55**
3040_8	**41**	48	53	56	66	66	66	66	**47**
3040_9	**43**	-	-	-	-	-	-	-	-
3040_10	**44**	-	-	-	-	-	-	-	-
3041_1	**35**	**37**	**37**	39	46	46	46	44	**37**
3041_2	**34**	40	39	44	43	43	43	48	**38**
3041_3	**40**	**41**	42	45	51	51	51	49	42
3041_4	**33**	**36**	37	37	49	49	49	43	37
3041_5	**37**	43	42	**38**	43	43	43	41	43
3041_6	**32**	37	37	**36**	44	44	44	44	37
3041_7	**39**	**43**	**43**	45	58	58	58	58	44
3041_8	**30**	**33**	**33**	34	41	41	41	40	34
3041_9	**42**	**42**	**42**	**42**	56	56	56	56	**42**
3041_10	**38**	40	**39**	43	47	47	47	47	41
3042_1	**35**	**38**	**38**	41	54	54	54	54	**38**
3042_2	**26**	28	28	**27**	32	32	32	35	28
3042_3	**37**	**37**	38	42	45	45	45	45	41
3042_4	**28**	**29**	30	31	31	31	31	30	30
3042_5	**35**	**36**	**36**	37	45	45	45	45	**36**
3042_6	**30**	**32**	**32**	33	39	39	39	39	**32**
3042_7	**29**	**32**	34	35	33	33	33	36	34
3042_8	**28**	31	**29**	30	34	34	34	34	**29**
3042_9	**31**	33	33	34	35	35	35	36	**32**
3042_10	**36**	38	**37**	39	39	39	39	39	38
3043_1	**29**	**30**	**30**	32	36	36	36	36	**30**
3043_2	**32**	**34**	**34**	35	41	41	41	41	**34**
3043_3	**26**	28	**27**	29	35	35	35	35	**27**
3043_4	**27**	**29**	**29**	32	33	33	33	33	**29**
3043_5	**44**	**44**	**44**	46	53	53	53	53	**44**
3043_6	**28**	**30**	32	31	40	40	40	41	**30**
3043_7	**33**	37	**34**	39	39	39	39	38	37
3043_8	**34**	**36**	37	39	41	41	41	41	37
3043_9	**38**	**40**	**40**	45	45	45	45	45	**40**
3043_10	**30**	33	**32**	34	40	40	40	36	33
3044_1	**41**	**43**	**43**	45	57	57	57	57	**43**
3044_2	**27**	29	**28**	29	37	37	37	37	**28**
3044_3	**34**	**35**	**35**	36	40	40	40	40	36
3044_4	**33**	35	35	35	38	38	38	38	**34**
3044_5	**29**	31	**30**	31	38	38	38	38	**30**
3044_6	**25**	28	**27**	**27**	37	37	37	37	**27**
3044_7	**35**	**36**	**36**	37	47	47	47	47	38
3044_8	**36**	38	39	41	49	49	49	49	**36**
3044_9	**35**	38	37	39	45	45	45	45	**36**
3044_10	**31**	**33**	**33**	35	35	35	35	35	**33**
3045_1	**40**	47	**45**	48	51	51	51	49	**45**
3045_2	**48**	54	55	56	**52**	**52**	**52**	57	57
3045_3	**42**	49	**48**	52	**48**	**48**	**48**	51	50
3045_4	**50**	57	**56**	59	58	58	58	58	58
3045_5	**42**	**45**	**45**	51	46	46	46	47	47
3045_6	**46**	**50**	51	57	57	57	57	54	52
3045_7	**46**	**48**	50	51	54	54	54	56	51
3045_8	**48**	54	55	57	56	56	56	**52**	53

benchmark		versões implementadas							
instancia	best	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
3045_9	**42**	**48**	**48**	52	55	55	55	51	49
3045_10	**42**	49	**47**	54	52	52	52	50	50
3046_1	**33**	36	36	36	42	42	42	40	**35**
3046_2	**40**	44	44	45	54	54	54	54	**43**
3046_3	**30**	36	35	38	44	44	44	41	**34**
3046_4	**37**	**42**	**42**	**42**	47	47	47	49	43
3046_5	**34**	**36**	37	42	42	42	42	43	37
3046_6	**33**	**38**	39	41	48	48	48	44	39
3046_7	**44**	**45**	**45**	50	50	50	50	49	48
3046_8	**33**	**37**	**37**	41	41	41	41	39	**37**
3046_9	**40**	**43**	**43**	46	50	50	50	50	45
3046_10	**37**	**40**	41	46	45	45	45	45	41
3047_1	**39**	40	41	40	48	48	48	48	**39**
3047_2	**29**	**30**	**30**	**30**	31	31	31	31	**30**
3047_3	**29**	32	32	**31**	37	37	37	33	32
3047_4	**38**	**39**	41	41	41	41	41	41	41
3047_5	**26**	30	**29**	30	30	30	30	30	**29**
3047_6	**34**	37	**36**	**36**	41	41	41	41	37
3047_7	**29**	31	31	31	**30**	**30**	**30**	**30**	**30**
3047_8	**38**	**38**	41	44	47	47	47	49	41
3047_9	**34**	36	36	37	40	40	40	39	**35**
3047_10	**42**	**42**	43	44	51	51	51	51	43
3048_1	**28**	30	30	33	32	32	32	32	**29**
3048_2	**30**	**32**	**32**	33	39	39	39	39	**32**
3048_3	**35**	**35**	37	39	41	41	41	41	**35**
3048_4	**24**	**26**	**26**	28	29	29	29	29	**26**
3048_5	**25**	**26**	**26**	27	33	33	33	33	27
3048_6	**31**	**32**	33	34	39	39	39	39	33
3048_7	**32**	34	34	36	45	45	45	45	**33**
3048_8	**28**	**29**	30	32	32	32	32	32	30
3048_9	**37**	**38**	40	42	42	42	42	42	40
3048_10	**28**	29	30	30	38	38	38	38	**28**
3049_1	**33**	**33**	35	35	40	40	40	35	36
3049_2	**49**	-	-	-	-	-	-	-	-
3049_3	**35**	**37**	38	44	38	38	38	38	**30**
3049_4	**37**	**38**	**38**	40	43	43	43	40	39
3049_5	**37**	**37**	38	43	43	43	43	42	38
3049_6	**31**	**34**	**34**	36	38	38	38	35	**34**
3049_7	**36**	**36**	**36**	37	41	41	41	38	**36**
3049_8	**34**	36	**34**	41	38	38	38	37	37
3049_9	**33**	**33**	35	38	36	36	36	36	34
3049_10	**30**	34	**33**	37	36	36	36	37	35
3050_1	**28**	**28**	**28**	30	34	34	34	31	**28**
3050_2	**37**	**37**	**37**	39	41	41	41	41	**37**
3050_3	**43**	**43**	**43**	44	47	47	47	47	**43**
3050_4	**34**	**34**	**34**	39	40	40	40	40	**34**
3050_5	**37**	**37**	**37**	**37**	**37**	**37**	**37**	39	**37**
3050_6	**37**	**37**	**37**	**37**	42	42	42	38	**37**
3050_7	**22**	**23**	**23**	**23**	34	34	34	29	**23**
3050_8	**27**	**28**	**28**	31	38	38	38	38	29
3050_9	**25**	26	**25**	27	29	29	29	29	28
3050_10	**30**	**30**	**30**	34	32	32	32	32	**30**
3051_1	**32**	**32**	**32**	**32**	37	37	37	37	**32**
3051_2	**23**	24	24	26	28	28	28	28	**23**
3051_3	**27**	**27**	**27**	29	33	33	33	33	28
3051_4	**51**	**51**	**51**	**51**	52	52	52	52	**51**
3051_5	**33**	**33**	**33**	34	48	48	48	48	34
3051_6	**38**	**38**	**38**	41	41	41	41	41	**38**
3051_7	**26**	**26**	**26**	**26**	30	30	30	30	**26**
3051_8	**33**	**33**	35	36	36	36	36	36	34
3051_9	**35**	**35**	**35**	38	45	45	45	45	**35**
3051_10	**27**	**27**	28	30	30	30	30	30	**27**
3052_1	**32**	**32**	**32**	34	36	36	36	36	**32**
3052_2	**23**	25	**24**	25	25	25	25	25	**24**
3052_3	**22**	**22**	**22**	24	24	24	24	24	**22**
3052_4	**29**	**29**	**29**	31	36	36	36	36	30
3052_5	**30**	**30**	**30**	**30**	38	38	38	38	**30**
3052_6	**30**	**30**	**30**	32	33	33	33	33	**30**
3052_7	**30**	**30**	**30**	**30**	31	31	31	31	**30**
3052_8	**26**	**26**	**26**	**26**	28	28	28	28	**26**
3052_9	**31**	32	**31**	**31**	32	32	32	32	**31**
3052_10	**29**	**29**	**29**	30	34	34	34	34	**29**
3053_1	**42**	46	**45**	51	50	50	50	47	47
3053_2	**38**	41	41	46	44	44	44	43	**39**
3053_3	**34**	**35**	37	37	36	36	36	36	36
3053_4	**44**	**48**	**48**	52	51	51	51	49	49
3053_5	**49**	55	**52**	56	59	59	59	53	55
3053_6	**43**	47	48	55	50	50	50	**44**	49
3053_7	**37**	43	**41**	44	46	46	46	42	42
3053_8	**39**	**43**	**43**	50	**43**	**43**	**43**	**43**	44
3053_9	**42**	48	48	53	52	52	52	48	**47**
3053_10	**37**	41	42	44	46	46	46	43	**40**
3054_1	**33**	**33**	**33**	36	36	36	36	36	**33**
3054_2	**28**	**29**	**29**	31	31	31	31	31	31
3054_3	**22**	**26**	**26**	28	28	28	28	**26**	27
3054_4	**30**	33	**32**	36	36	36	36	33	34
3054_5	**28**	**29**	**29**	34	**29**	**29**	**29**	**29**	31
3054_6	**30**	32	**31**	**31**	32	32	32	32	32
3054_7	**39**	40	40	40	41	41	41	**39**	**39**
3054_8	**31**	**33**	**33**	34	35	35	35	35	34
3054_9	**28**	33	32	33	32	32	32	**31**	**31**
3054_10	**29**	**33**	**33**	36	34	34	34	34	34
3055_1	**29**	**29**	**29**	30	39	39	39	36	**29**
3055_2	**31**	33	**31**	36	33	33	33	36	32
3055_3	**28**	**29**	**29**	31	32	32	32	32	**29**
3055_4	**41**	**41**	**41**	**41**	**41**	**41**	**41**	**41**	**41**

benchmark		versões implementadas							
instancia	best	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
3055_5	**24**	26	**25**	28	29	29	29	28	26
3055_6	**25**	**26**	**26**	**26**	28	28	28	28	**26**
3055_7	**44**	**44**	**44**	45	47	47	47	47	**44**
3055_8	**44**	**44**	**44**	**44**	51	51	51	51	**44**
3055_9	**27**	**27**	**27**	**27**	28	28	28	**27**	**27**
3055_10	**29**	**30**	**30**	**30**	36	36	36	36	31
3056_1	**21**	24	**23**	25	**23**	**23**	**23**	**23**	**23**
3056_2	**31**	33	34	35	35	35	35	35	**32**
3056_3	**34**	**34**	**34**	**34**	38	38	38	38	**34**
3056_4	**32**	**32**	**32**	**32**	34	34	34	34	**32**
3056_5	**29**	**29**	**29**	30	36	36	36	36	**29**
3056_6	**26**	**26**	**26**	29	27	27	27	27	**26**
3056_7	**30**	**30**	**30**	**30**	32	32	32	32	**30**
3056_8	**29**	**29**	**29**	35	40	40	40	40	**29**
3056_9	**33**	34	35	38	38	38	38	38	**33**
3056_10	**39**	**39**	**39**	**39**	**39**	**39**	**39**	**39**	**39**
3057_1	**30**	31	31	36	31	31	31	33	**30**
3057_2	**32**	**35**	**35**	42	39	39	39	**35**	36
3057_3	**29**	**29**	**29**	33	**29**	**29**	**29**	**29**	**29**
3057_4	**29**	30	**29**	31	30	30	30	**29**	**29**
3057_5	**27**	**27**	**27**	30	**27**	**27**	**27**	**27**	**27**
3057_6	**23**	**24**	**24**	25	25	25	25	25	25
3057_7	**27**	27	28	33	**27**	**27**	**27**	**27**	29
3057_8	**29**	**29**	31	31	31	31	31	**29**	31
3057_9	**26**	**26**	**26**	28	27	27	27	28	**26**
3057_10	**32**	33	33	33	**32**	**32**	**32**	**32**	33
3058_1	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
3058_2	**32**	33	**32**	33	**32**	**32**	**32**	**32**	**32**
3058_3	**31**	32	32	32	**31**	**31**	**31**	**31**	**31**
3058_4	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**
3058_5	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
3058_6	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**
3058_7	**26**	28	28	28	27	27	27	**26**	27
3058_8	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
3058_9	**30**	33	32	32	**30**	**30**	**30**	31	31
3058_10	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**
3059_1	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**
3059_2	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
3059_3	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**
3059_4	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
3059_5	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
3059_6	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**
3059_7	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**
3059_8	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**26**
3059_9	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**
3059_10	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3060_1	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**
3060_2	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**38**
3060_3	**39**	**39**	**39**	**39**	**39**	**39**	**39**	**39**	**39**
3060_4	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
3060_5	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**
3060_6	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**
3060_7	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
3060_8	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3060_9	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
3060_10	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
3061_1	**37**	42	41	42	41	41	41	**39**	**39**
3061_2	**36**	39	39	39	39	39	39	**38**	**38**
3061_3	**39**	45	45	45	**42**	**42**	**42**	**42**	43
3061_4	**38**	43	43	47	43	43	43	**40**	**40**
3061_5	**39**	44	44	47	44	44	44	**43**	**43**
3061_6	**35**	**38**	**38**	40	**38**	**38**	**38**	**38**	**38**
3061_7	**44**	**47**	48	52	49	49	49	**47**	51
3061_8	**47**	52	**50**	55	53	53	53	52	52
3061_9	**44**	49	50	56	52	52	52	50	**46**
3061_10	**41**	**45**	46	49	46	46	46	46	46
3062_1	**26**	**28**	29	33	31	31	31	**28**	**28**
3062_2	**33**	35	35	35	35	35	35	**34**	**34**
3062_3	**26**	28	28	31	29	29	29	**26**	27
3062_4	**33**	**33**	**33**	36	**33**	**33**	**33**	**33**	**33**
3062_5	**42**	**42**	**42**	**42**	**42**	**42**	**42**	**42**	**42**
3062_6	**36**	40	40	40	40	40	40	**38**	**38**
3062_7	**27**	**30**	**30**	33	**30**	**30**	**30**	**30**	**30**
3062_8	**38**	39	39	39	39	39	39	**38**	**38**
3062_9	**26**	27	27	27	**26**	**26**	**26**	**26**	**26**
3062_10	**26**	28	28	28	**26**	**26**	**26**	**26**	**26**
3063_1	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3063_2	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
3063_3	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**
3063_4	**31**	33	33	35	33	33	33	**31**	**31**
3063_5	**37**	**37**	**37**	**37**	**37**	**37**	**37**	**37**	**37**
3063_6	**39**	**39**	**39**	**39**	**39**	**39**	**39**	**39**	**39**
3063_7	**26**	27	27	27	27	27	27	**26**	**26**
3063_8	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
3063_9	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
3063_10	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**
3064_1	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
3064_2	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
3064_3	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**
3064_4	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
3064_5	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**
3064_6	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**	**29**
3064_7	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**
3064_8	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
3064_9	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
3064_10	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**

Tabela 6: Resultados das oito versões da implementação vs benchmark (cont.)

Tendo sido implementado em oito versões (tabela 4), diferenciadas na aplicação da BT quanto à varredura da vizinhança, conforme descrito na seção anterior, o trabalho apresentou os resultados finais (best) que foram elencados nas tabelas 5 e 6. Nas duas primeiras colunas (benchmark) encontramos o nome da instância e o melhor makespan, encontrado por outros autores; nas 8 colunas seguintes (V1, V1.a, V1.b, V2, V2.a, V2.b, V3 e V4) encontramos os resultados finais obtidos, respectivamente, pelas 8 versões da implementação. A ausência de valor em um par versão e instância indica que a versão implementada não encontrou uma solução viável para aquela instância, isto é, a solução apresentou um makespan maior que o tempo máximo e/ou violou a disponibilidade de recursos não renováveis. Os valores negritados correspondem ao melhor resultado encontrado entre as oito versões para aquela instância; quando, além disso, igualarem ao do benchmark correspondente, os valores são destacado em fundo cinza.

As 70 primeiras instâncias, de números 301_1 a 307_10, apesar de incluídas no conjunto das 640 processadas e de terem sido computadas nos números dos resultados finais, não foram apresentadas nas tabelas 5 e 6 por economia de espaço, pois também não se encontraram resultados viáveis para estas instâncias.

As tabelas 7 a 10 apresentam o resumo não somente dos resultados finais, como também das fases intermediárias, de Solução Inicial (SI), de Busca Tabu (BT) e de Reconexão de Caminhos (RC), relacionando com as versões da implementação.

	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
Makespan iguais ao benchmark	**236(44,4%)**	224(42,1%)	175(32,9%)	180(33,8%)	180(33,8%)	180(33,8%)	204(38,4%)	**244(46,0%)**
Makespan maiores que o benchmark	**296(55,6%)**	308(57,9%)	357(67,1%)	352(66,2%)	352(66,2%)	352(66,2%)	327(61,6%)	**287(54,0%)**
Quantidade de makespan obtidos	532	532	532	532	532	532	531	531

Tabela 7 – Resultados finais obtidos nas oito versões (640 instâncias)

Conforme a tabela 7, apesar de não encontrar *makespan* inferior ao do benchmark, foram obtidos, em média, 532 resultados, ou seja, 96,2% do universo de 552 soluções existentes. Entretanto, muitos destes resultados obtidos foram acima do benchmark (de

54% a 67% dos resultados de cada implementação). Entre as soluções encontradas, destacam-se as obtidas pelas versões V1 e V4, cujas quantidades de soluções iguais à do benchmark correspondem a 44,4 % e 46,0 % dos respectivos totais de soluções viáveis alcançadas.

	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
resultado BT melhorado por RC	**239**	**239**	174	0	0	0	2	**182**
resultado BT melhorado por RC e igualado ao benchmark	**56**	**45**	40	0	0	0	0	**42**

Tabela 8 – Resultados da RC em relação aos da BT

Pela tabela 8, a Reconexão de Caminhos (RC) melhorou um bom número de resultados encontrados pela Busca Tabu (BT), sendo 239 nas versões V1 e V1.a e 182 na versão V4 do código. Considerando os totais de soluções viáveis obtidas, a RC melhorou 45% (V1 e V1.a), 34% (V4) e 33% (V1.a). Já as versões V2, V2.a, V2.b e V3 não apresentaram qualquer melhoria, e o motivo deve estar aliado à principal diferença deste grupo de versões, que é o uso da BT sem heurística gulosa na troca de modos.

	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
resultado BT igual ao benchmark	180	179	158	180	180	180	**204**	**202**
resultado BT acima do benchmark	352	353	374	352	352	352	**327**	**329**

Tabela 9 – Resultados da BT comparados ao benchmark

Ainda pelas diferenças de implementação, as versões V1 e V2, que não utilizaram o operador de troca de ordem das atividades nas vizinhanças da BT, apresentaram resultado da BT menos aderente ao benchmark, conforme a tabela 9: os percentuais de resultados iguais aos do benchmark foram de 33,8 % (V1, V1.a e grupo V2) e de 38,0 % (V3 e V4) sobre os totais de soluções viáveis encontradas.

	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
resultado SI não viável melhorado por BT	**14**	**14**	**14**	**14**	**14**	**14**	13	13
resultado SI viável melhorado por BT	**303**	**309**	244	268	268	268	**293**	**348**
Totais dos resultados SI melhorados	**317**	**323**	258	282	282	282	**306**	**361**

Tabela 10 – Resultados da Solução Inicial em relação aos da BT

Lembrando que, quanto à geração das Soluções Iniciais (SI) da BT, foi introduzido um balanceamento dos modos iniciais das atividades, buscando equilibrar a redução do tempo de duração (e do makespan) com a demanda de recursos, e que, em seguida, foi aplicada uma busca tabu com o mesmo objetivo (viabilizar tempo e recursos), foi confirmado o bom desempenho geral da Busca Tabu. Conforme a tabela 10, ainda que partindo de uma SI não viável, a BT proporcionou melhoria em torno de 13 instâncias. Sobre a totalidade de resultados encontrados, a BT melhorou 59,6% (V1), 60,71%(V1.a), 48,49% (V1.b), 53,0% (grupo V2), 57,6% (V3) e 68,0% (V4) das soluções iniciais elaboradas.

Assim, as versões V1 e V4 apresentaram melhor desempenho no alcance de resultados aderentes ao benchmark (tabela 7). Porém, na BT, ainda em comparação ao benchmark, destacaram-se as versões V3 e V4 (tabela 9). A melhoria de desempenho, na versão V1, ocorreu com a RC, que produziu melhorias da BT na ordem de 45% das soluções Tabu encontradas (tabela 8).

	V1	**V1.a**	**V1.b**	**V2**	**V2.a**	**V2.b**	**V3**	**V4**
resultado BT pior que os das demais versões	**226**	245	353	296	296	296	**213**	**177**
resultado BT melhor que os das demais versões	**37**	30	2	0	0	0	**44**	**38**
resultado BT igual à melhor solução das demais versões	**269**	257	177	236	236	236	**274**	**316**
totais	532	532	532	532	532	532	531	531

Tabela 11 – Resultados da BT comparativos entre as oito versões

Comparando-se os resultados da BT obtidos pelas versões (tabela 11) da implementação, conclui-se que a versão V3 apresentou melhores resultados, pois superou as demais em 44 instâncias, ficando empatadas na segunda colocação as versões V1 e V4, com 37 e 38 instâncias. A versão V4 apresentou melhor robustez, avaliando-se a produção de resultados piores que os das demais versões, com apenas 177, contra 226 e 213 das versões V1 e V3.

A dispersão e a avaliação dos resultados que divergiram do benchmark estão apresentados nas tabelas 12, 13 e 14.

Com base nas soluções obtidas, foram apurados alguns dados estatísticos dos percentuais de distanciamento (desvio) das soluções do benchmark, separados por

versão do código. Buscando-se avaliar segmentos separados do universo de soluções obtidas, os resultados foram assim agrupados: a tabela 12 contempla apenas os resultados que divergiram do benchmark; por sua vez, a tabela 13 reúne todas as soluções viáveis alcançadas, iguais ou não ao benchmark.

	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
menor percentual de desvio	2,30%	2,27%	2,27%	1,96%	1,96%	1,96%	1,96%	2,22%
maior percentual de desvio	44,70%	30,95%	36,84%	60,98%	60,98%	60,98%	60,98%	29,79%
média dos afastamentos	**9,72%**	**9,49%**	13,58%	17,71%	17,71%	17,71%	16,59%	**9,18%**
desvio padrão dos afastamentos	**5,86%**	**5,35%**	7,18%	11,57%	11,57%	11,57%	11,62%	**4,93%**

Tabela 12 – Percentuais de afastamento das soluções que divergiram do benchmark

	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
média dos afastamentos	**5,41%**	**5,49%**	9,11%	11,72%	11,72%	11,72%	10,22%	**4,96%**
desvio padrão dos afastamentos	**6,52%**	**6,21%**	8,68%	12,60%	12,60%	12,60%	12,18%	**5,84%**

Tabela 13 – Percentuais de afastamento de todas as soluções em relação ao benchmark

Podemos verificar, que a solução da V4 supera as demais, porém próxima às versões V1 e V1.a, com menor média e o menor desvio padrão, tanto sobre o grupo de soluções divergentes do benchmark (tabela 12) quanto no grupo da totalidade soluções viáveis obtidas, divergentes ou não do benchmark (tabela 13). As versões do grupo V2 e a V3 apresentam médias altas, e a redução destas médias pela diluição decorrente da inclusão dos resultados idênticos ao benchmark, (da tabela 12 para a tabela 13), é razoavelmente inferior à alcançada nas V1 e V4. Os desvios padrões altos, principalmente nas versões do grupo V2 e na V3, indicam uma dispersão alta, que de fato ocorre.

%	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
0	236	224	175	180	180	180	204	244
10	169	180	128	105	105	105	112	168
20	112	115	155	115	115	115	106	112
30	12	11	69	80	80	80	60	7
40	2	2	5	33	33	33	33	0
50	1	0	0	14	14	14	12	0
>50	0	0	0	5	5	5	4	0

Tabela 14 – Distribuição dos percentuais de afastamento nas oito versões

Para visualizar melhor esta dispersão, temos na tabela 14 a distribuição destes percentuais por faixas. As versões V1, V1.a e V4 concentram seus desvios nas faixas menores, confirmando a média, de distanciamento dos resultados do benchmark, próxima a 5%. Novamente, as versões 1 e 4 são evidenciadas com melhores resultados que as demais.

	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4
tempo médio	14 s	15 s	14 s	13 s	40 s	40 s	04:22 min	04:24 min
tempo máximo	18 s	33 s	28 s	17 s	50 s	58 s	08:20 min	09:16 min
desvio padrão	1s	2 s	1 s	1s	3 s	4 s	01:19 min	01:42 min

Tabela 15 – Estatística dos tempos de processamento

Verificando os tempos individuais médios de processamento das versões, conforme a tabela 15, podemos verificar que a diferença do tempo de processamento das versões dos grupos V1 e V2, comparadas às versões V3 e V4, é muito grande. De uma média de 14 segundos das versões V1, V1.a, V1.b e V2, sobe-se para um 40 segundos nas versões V2.a e V2.b, e, muito mais, para 4:24 minutos nas versões V3 e V4. Desta forma, a versão V1 se destaca não só das versões 2 e 3, como verificado nas avaliações anteriores, como também da versão V4, pelo tempo reduzido de processamento.

instância	bloco	V1	V1.a	V1.b	V2	V2.a	V2.b	V3	V4	melhor makespan
3011_1	INICIO	01:53:19	00:47:31	00:19:46	09:37:51	01:23:42	08:35:20	18:33:17	09:13:44	Makespan igual ao benchmark na versão V4 e no grupo V1
	BT	01:53:19	00:47:31	00:19:46	09:37:51	01:23:42	08:35:20	18:33:17	09:13:44	
	RC	01:53:33	00:47:45	00:19:59	09:38:04	01:24:20	08:36:00	18:37:28	09:17:59	
	FIM	01:53:33	00:47:45	00:19:59	09:38:04	01:24:20	08:36:00	18:37:28	09:17:59	
	total (min.)	**00:14**	**00:14**	**00:13**	**00:13**	00:38	00:40	04:11	04:15	
308_6	INICIO	01:47:26	00:41:26	00:13:58	09:32:22	01:07:08	08:18:07	04:33:54	21:14:51	Makespan igual ao benchmark na versão V4 e no grupo V1
	BT	01:47:26	00:41:26	00:13:58	09:32:22	01:07:08	08:18:07	04:33:54	21:14:51	
	RC	01:47:40	00:41:41	00:14:11	09:32:36	01:07:49	08:18:49	04:38:16	21:19:16	
	FIM	01:47:40	00:41:41	00:14:11	09:32:36	01:07:49	08:18:49	04:38:16	21:19:16	
	total (min.)	**00:14**	**00:15**	**00:13**	**00:14**	00:41	00:42	04:22	04:25	
307_8	INICIO	01:45:36	00:39:31	00:12:06	09:30:32	01:01:34	08:12:23	03:59:31	20:25:23	Melhor makespan na versão V4 (porém acima do benchmark)
	BT	01:45:36	00:39:31	00:12:06	09:30:32	01:01:34	08:12:23	03:59:31	20:25:23	
	RC	01:45:49	00:39:46	00:12:20	09:30:45	01:02:16	08:13:06	04:03:46	20:29:34	
	FIM	01:45:49	00:39:46	00:12:20	09:30:45	01:02:16	08:13:06	04:03:46	20:29:34	
	total (min.)	**00:13**	**00:15**	**00:14**	**00:13**	00:42	00:43	04:15	04:11	
309_8	INICIO	01:50:18	00:44:25	00:16:47	09:35:03	01:15:16	08:25:09	05:27:43	22:22:36	Melhor makespan nas versões V1 e V1.a (porém acima do benchmark)
	BT	01:50:18	00:44:25	00:16:47	09:35:03	01:15:16	08:25:09	05:27:43	22:22:36	
	RC	01:50:32	00:44:40	00:17:01	09:35:15	01:15:54	08:25:51	05:31:58	22:31:19	
	FIM	01:50:32	00:44:40	00:17:01	09:35:15	01:15:54	08:25:51	05:31:58	22:31:19	
	total (min.)	**00:14**	**00:15**	**00:14**	**00:12**	00:38	00:42	04:15	08:43	

Tabela 16 – Exemplos de tempos de processamento individuais

A tabela 16 apresenta, para cada versão da implementação, os tempos de processamento sobre algumas instâncias. A primeira coluna identifica a instância e, da terceira à décima colunas, apresenta a hora correspondente ao instante de início de cada etapa do programa (Inicio, Busca Tabu, e Reconexão de Caminhos) indicados na segunda coluna, além do instante de término (Fim). Para cada versão e respectiva instância, também é indicado o tempo total de processamento em minutos. Na última coluna, indica características dos makespan obtidos. Os exemplos da tabela 16 correspondem às médias de tempo apuradas na tabela 15. É importante observar que o tempo de processamento, das soluções implementadas, é decorrente, quase que exclusivamente, do tempo da Busca Tabu, à exceção somente dos décimos de segundo dispendidos na Solução Inicial e na Reconexão de Caminhos. Daí a importância do desempenho da Busca Tabu executada sobre a vizinhança, em relação aos algoritmos da Solução Inicial (também com uma Busca Tabu) e da Reconexão de Caminhos, para o comportamento geral da implementação.

Os resultados observados até aqui destacam a eficácia da versão gulosa da BT e demonstram que a *troca de ordem* determina alguma melhora nos resultados, porém penaliza, excessivamente, o tempo de processamento.

Apesar dos resultados alcançados, próximos ao benchmark, alguns eventos não corresponderam à expectativa. Além das 88 instâncias sem solução conhecida, as versões implementadas deixaram de obter solução viável para outras 19 instâncias.

No benchmark, estas instâncias foram geradas com parâmetros diversos, propiciando uma razoável diversidade na complexidade de cada uma. Esta diversidade impacta no espaço solução, ampliando ou reduzindo. Também o tempo na obtenção de bons resultados torna-se muito diferente entre as instâncias, podendo ser imediato ou estendido. Algumas instâncias apresentaram, invariavelmente, soluções não viáveis (dificuldade nas restrições de tempo máximo e de recursos não renováveis), conforme observado no benchmark. Essa heterogeneidade entre as instâncias pode ser exemplificada: o resultado igual ao benchmark na instancia 3011_1 foi sempre rapidamente alcançado; a instância 307_8 apresentou um nível médio de dificuldade; já a instância 308_6, por sua vez, determinou um nível elevado de dificuldade na obtenção de resultados viáveis. Outra característica importante foi a diferença dos resultados obtidos, em decorrência de variações nos critérios de pesos da lista de modos na solução inicial. Dependendo deste balanceamento e da instância, a solução inicial já se aproximava da solução igual ao benchmark ou, simetricamente, se distanciava muito.

Destacando-se entre as versões, a V4 incluiu em seu código a Busca Tabu com troca de ordem das atividades e a heurística gulosa na troca de modos da vizinhança tabu. Em decorrência, apresentou robustez nos resultados, tanto no Tabu quanto no resultado final, após a RC. As variações da Busca Tabu nas versões V1 (V1.a e V1.b) e V2 (V2.a e V2.b), com a Lista Tabu apresentando Prazo Tabu de 30 (V1.a e V2.a) e 1500 (V1.b e V2.b) ciclos, acrescentaram pouca diferença. A V1 apresentou melhor resultado, apesar de implementar uma Busca Tabu adaptada (com Prazo Tabu infinito). A versão com Prazo Tabu menor (30 ciclos) comportou-se melhor que a outra, com prazo muito grande (1500 ciclos). Por sua vez, as variações da V2 não apresentaram qualquer variação nos resultados, sempre muito próximos daqueles de V2. Apenas ampliaram o tempo de processamento.

A Reconexão de Caminhos apresentou um ganho significativo nas soluções finais a partir das soluções elite do tabu, associada à BT com heurística gulosa, utilizadas pelas versões V1, V1.a, V1,b e V4.

5. Conclusão

O Problema de Escalonamento de Projetos com Restrição de Recursos vem sendo amplamente avaliado em diferentes abordagens, inclusive metaheurísticas, por vários autores [26].

Neste trabalho, buscou-se avaliar o RCPSP, em sua versão multimodo, através da metaheurística Busca Tabu, intensificada pelo uso da Reconexão de Caminhos como estratégia de pós-otimização. Procurou-se estabelecer uma comparação da varredura da vizinhança por formas distintas, avaliando-se os resultados produzidos. O resultado, sobretudo da intensificação, mostrou-se interessante e constituído de boas soluções.

Para a implementação da Busca Tabu, foram empregados alguns elementos utilizados por Nonobe e Ibaraki [35][34]. Para a estruturação das soluções, utilizou-se esquemas de pares de listas (m,l), que correspondem, respectivamente, à lista de modos – modos que foram escolhidos para a execução de cada atividade – e à lista de atividades – seqüência de escalonamento das atividades no cronograma do projeto. Esta estrutura de representação das soluções do RCPSP, que proporcionam o alicerce do trabalho das heurísticas, apresentou-se eficiente, demonstrando ser uma boa representação para os resultados deste problema, conforme avaliado por outros autores em seus trabalhos [26].

O algoritmo da Busca Tabu, com os movimentos de *troca de modos* e *troca de ordem* das atividades, sobre as estruturas de representação do problema, respectivamente, a lista de modos e lista de atividades, gerou soluções interessantes. Entretanto, apesar da *troca de ordem* ter ampliado o espaço de busca e proporcionado certa melhoria, proporcionou um desempenho bastante inferior ao da *troca de modos*, e implicou numa ampliação muito significativa no tempo de processamento. Por fim, a estratégia de geração da Solução Inicial, desvinculando os critérios para o estabelecimento da lista de modos (esta resultante de uma BT, para promover viabilidade e uma pequena otimização) e da lista de atividades da SI, tornou-a bastante diversificada.

Buscando uma mudança na tática de varredura da vizinhança da BT, se estabeleceu uma heurística gulosa, que foi especificamente empregada no operador *troca de modos* da lista modos. Por sua vez, a *troca de ordem* sobre a lista de atividade apresentou pequena atuação sobre a melhoria dos resultados, determinando um questionamento sobre a sua eficiência. Sobre a estruturação da Lista Tabu na *troca de modos*, foram geradas

implementações com Prazos Tabu distintos: numa primeira versão adaptada, utilizou-se Prazo Tabu infinito (apoiado em uma lista dinâmica), e nas duas versões seguintes, com a Busca Tabu em sua forma canônica, utilizaram-se Prazos Tabu de 30 e de 1500 ciclos.

Deste modo, foram implementadas oito versões distintas, para efeito de comparação destas estratégias de varredura na vizinhança da Busca Tabu: com e sem *troca de ordem*, combinadas com *troca de modo* gulosa e não gulosa, além de Prazos Tabu significativamente distintos. Os resultados demonstraram eficiência na versão gulosa da BT e confirmaram que, apesar de pouca, a *troca de ordem* determina alguma melhora nos resultados, apesar de ampliar, em muito, o tempo computacional. Além disso, verificou-se que a Busca Tabu adaptada, com Prazo Tabu infinito, na *troca de modo* gulosa, indicou melhor desempenho que aquelas com Prazo Tabu definidos.

Ainda que a Busca Tabu tenha apresentado interessantes resultados viáveis, para quase todas as instâncias com resultados conhecidos (96,1%), ocorreu um enriquecimento qualitativo com o uso da Reconexão de Caminhos, evoluindo as soluções elite para resultados próximos ou iguais ao do benchmark. Optou-se por uma economia no tempo de processamento, avaliando-se somente pares de soluções consecutivas (valores próximos), pois apresentam menor diversidade de soluções intermediárias a serem avaliadas. Desta forma, confirmou-se a eficiência da RC em promover melhorias nas soluções obtidas em metaheurísticas, como a BT.

Um trabalho futuro constitui-se em intensificar a investigação e entender por que não foram encontradas soluções viáveis em algumas instâncias.

Em próximos trabalhos, a Busca Tabu pode ser enriquecida, estabelecendo-se outros critérios para a Lista Tabu – com tamanho dinâmico e determinando-se um limite de iterações sem melhorias de resultados.

Quanto à Reconexão de Caminhos, pode-se reconsiderar o procedimento, com base em outras formas de composição [30][40][19]. Pode ser implementada como estratégia de intensificação dos ótimos locais, a partir de um repositório contendo estes valores obtidos no decorrer da Busca Tabu, e não somente ao final, sobre as soluções elite. Também na RC, é possível estabelecer a solução inicial como a melhor do par a ser avaliado, deixando para a solução guia a de resultado inferior, buscando intensificar a busca na vizinhança do resultado mais promissor. Deve-se, além disso, para ampliar o

espaço de busca, estendendo as comparações a todos os pares de soluções do repositório a ser avaliado.

Acerca da Solução Inicial, é interessante reavaliar a estratégia de geração, mais no que se refere à construção da lista de modos (viável e otimizada), do que na elaboração da lista de atividades. Esta desconexão na geração de ambas promove uma diversificação no espaço solução, e avaliou-se que variações consecutivas da SI implicaram em alterações significativas dos resultados. Também, aplicando-se algumas modificações, especificamente no critério de montagem da lista de modos, produziu-se, para algumas instâncias, soluções viáveis e melhores que as obtidas nas versões da implementação apresentadas neste trabalho.

Outro interessante estudo futuro é o de verificar a correspondência entre os resultados de melhoria proporcionados pela RC e a aplicação da heurística gulosa na *troca de modos* operada na vizinhança da Busca Tabu, uma vez que somente nestes casos a Reconexão de Caminhos promoveu evolução de resultados. De imediato, podemos descartar a possibilidade de falta de qualidade nas soluções da BT com heurística gulosa. É importante lembrar que, quando comparadas entre si, estas versões da BT proporcionaram mais robustez que aquelas sem a referida heurística, pois apresentaram freqüências maiores de soluções melhores ou iguais às soluções das demais versões.

REFERÊNCIAS

[1] T. Baar, P. Brucker, S. Knust, "**Tabu-Search Algorithms for the Resource-Constrained Project Scheduling Problem**", in: S. Voss, S. Martello, I. Osman, C. Roucairol (eds): Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academics Publishers (1988) 1-18

[2] J. Blazewicz, J.K. Lenstra, A.H.G. Rinooy Kan, "**Scheduling subject to resource constraints: Classification and Complexity**", Discrete Applied Mathematics, 5:11-24, 1983

[3] K. Bouleimen, H.. Lecocq, "**A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem**", European Journal of Operational Research 149: 268-281, 2003

[4] P. Brucker, S. Knust, "**Lower bounds for resource-constrained project scheduling problems**", European Journal of Operational Research 149: 302-313, 2003

[5] C. C. B. Cavalcante, C. C. Ribeiro, V. C. CAVALCANTE, C. C. SOUZA, **"Parallel Cooperative Approaches for the Labor Constrained Scheduling Problem**", In: Celso da Cruz Carneiro Ribeiro; Pierre Hansen. (Org.). Essays and Surveys in Metaheuristics. Boston: Kluwer Academic Publishers, 2001, v. , p. 201-225.

[6] C. C. B. Cavalcante, C. C. Souza, "**A Tabu Search Approach for Scheduling Problem under Labor Constraints**", Technical Report IC-97-13, ICUNICAMP, 1997

[7] C. C. B. Cavalcante, C. C. de Souza, L. A. Wolsey, "**Scheduling Projects with Labor Constraints**", Discrete Applied Mathematics Journal, vol 112, number 1-3, 27-52, 2001

[8] J.S. Coelho, L.V. Tavares, "**Comparative Analysis on approximation algorithms for the Resource Constrained Project Scheduling Problem**", PMS2002 (Eighth International Workshop on Project Management and Scheduling), http://jcoelho.m6.net/edicao3.asp?pa=4162

[9] D. Debels, B. Reyck, R. Leus, M. Vanhoucke, "**A Hybrid Scatter Search / Electromagnetism Meta-Heuristic for Project Scheduling**", European Journal of Operational Research, vol. 169, no. 2 (Mar.), 638 - 653, 2006

[10] T. M. Dias, D. F. Ferber, "**Tabu e Algoritmos Genéticos aplicados a problemas de escalonamento**", XXXIII Simpósio Brasileiro de Pesquisa Operacional, 1624-1632, ISSN 1518-1731, 2001

[11] K. Fleszar, K. S. Hindi, "**Solving the resource-constrained project scheduling problem by a variable neighbourhood search**", European Journal of Operational Research, Vol. 155, pp. 402–413, 2004

[12] F. Glover, "**Tabu Search — Part I**", *ORSA Journal on Computing*, *1*: 3, 190-206, 1989

[13] F. Glover, "**Tabu Search — Part II**", *ORSA Journal on Computing*, *2*: 1, 4-32, 1990

[14] F. Glover, "**Tabu Search and adaptive memory progamming – advances, applications and challenges**", in Barr, Hegalson, Kennington, (Ed). Interfaces in Computer Science and Operations Research. Boston Dordrecht London: Kluwer, 1-75, 1996

[15] F. Glover, M. Laguna, "**Tabu Search**". Boston Dordrecht London: Kluwer Academic Publishers, 1997

[16] S. Hartmann, "**A self-adaptive genetic algorithm for project scheduling under resource constraints**", Naval Research Logistics, 49:433-448, 2002

[17] S. Hartmann, "**A competitive genetic algorithm for the resource-constrained project scheduling**", Naval Research Logistics, 45:733--750, 1998

[18] R. Heilmann, "**Resource-constrained project scheduling: a heuristic for the multi-mode case**", OR Spektrum 23:335-357, 2001

[19] S.C. Ho, M. Gendreau, "**Path relinking for the vehicle routing problem**", Journal of Heuristics, 12: 55–72, 2006

[20] J. A. Ichihara, "**Um método de solução heurístico para a programação de edifícios dotados de múltiplos pavimentos-tipo**", Tese de Doutorado, EPS/UFSC, 1998

[21] R. Kolisch, "**Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation**", European Journal of Operational Research, 90:320-333, 1996

[22] R. Kolisch, "**Project Scheduling under resource constraints – Efficient heuristics for several problem classes**", Springer-Verlag, Heidelberg, 212 , 1995

[23] R. Kolisch, A. Drexl, "**Adaptive search for solving hard project scheduling problems**", Naval Research Logistics, 43:23-40, 1996

[24] R. Kolisch, S. Hartmann, "**Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis**", Project Scheduling: Recent Models, Algorithms and Applications, 147-178, Kluwer, Amsterdam, the Netherlands, 1999

[25] R. Kolisch, S. Hartmann, "**Experimental Evaluation of State-of-the-Art Heuristics for the Resource-Constrained Project Scheduling Problem**", European Journal for Operational Research, Vol. 127, 2, 394-407, 2000

[26] R. Kolisch, S. Hartmann, "**Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling: An Update**", European Journal of Operational Research, vol 174, 1, 23-37, 2005

[27] R. Kolisch, R. Padman, "**An Integrated Survey of Project Scheduling**", Tech. Rep. 463, University of Kiel, 1997

[28] R. Kolisch, A. Sprecher, "**Benchmark instances for project scheduling problems**", Handbook on recent advances in project scheduling. Chapter 9, J. Weglarz(ed.) Kluwer, Dordrecht. Universidade de Kiel, Alemanha, 197-212, 1998

[29] V. J. Leon, B. Ramamoorthy, "**Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling**", OR Spektrum, 17(2/3):173-182, 1995

[30] E. H. Marinho, "**Busca Tabu aplicada ao Problema de Programação de Tripulações de ônibus Urbano**"; XXXVI Simpósio Brasileiro de Pesquisa Operacional; 1;1471- 1482; 2004

[31] D. Merkle, M. Middendorf, H. Schmeck, "**Ant Colony Optimization for Resource-Constrained Project Scheduling**", Proceedings of the Genetic and Evolutionary Computation Conference -2000, 893-900, 2000

[32] K. Naphade, S. Wu, R. Storer, "**Problem space search algorithms for resource-constrained project scheduling**", Annals of Operations Research, 70:307-326, 1997

[33] K. Neumann, C. Schwindt, J. Zimmermann, "**Project Scheduling with time windows and scarce resources: temporal and resource-constrained project scheduling with regular and nonregular objective functions**", Springer, 2002

[34] K. Nonobe, T. Ibaraki, "**Tabu Search Algorithm for a Generalized Resource Constrained Project Scheduling Problem**", MIC – 5th Metaheuristics International Conference, Kyoto, August 25-28, 2003

[35] K. Nonobe, T. Ibaraki, "**Formulation and Tabu Search Algorithm for the Resource Constrained Project Scheduling Problem**", in: C.C. Ribeiro and P. Hansen (eds.): Essays and Surveys in Metaheuristics, Kluwer Academic Publishers, pp.557-588, 2002

[36] K. Nonobe, T. Ibaraki, "**A Local Search Approach to the Resource Constrained Project Scheduling Problem to Minimize Convex Costs**", MIC – 4th Metaheuristics International Conference, 75-78, 2001

[37] K. Nonobe, T. Ibaraki, "**A Tabu Search Approach to the constraint satisfaction problem as a general problem solver**", European Journal of Operational Research, 106, 599-623, 1998

[38] **PMBOK® Guide** , Project Management Institute, A guide to the project management body of knowledge, 2000

[39] B. De Reyck, W. Herroelen, "**The multi-mode resource-constrained project scheduling problem with generalized precedence relations**", European Journal of Operational Research 119: 538-556, 1999

[40] C. C. Ribeiro, E. UCHOA, R.F. WERNECK, "**A Hybrid GRASP with perturbations for the Steiner problems in graphs**", INFORMS Journal on Computing, Linthicum, v. 14, 228-246, 2002

[41] I. C. M. Rosseti, "**Estratégias seqüenciais e paralelas de GRASP com reconexão por caminhos para o problema de síntese de redes a 2-caminhos**"**,** Tese de Doutorado, PUC, Rio de Janeiro, 2003

[42] Schwindt, "**Generation of resource-constrained project scheduling problems subject to temporal constraints**", Report WIOR-543, Universidade de Karlsruhe, 1998

[43] V. Valls, S. Quintanilla, F. Ballestin, "**Resource-constrained project scheduling: a critical activity reordering heuristic**", European Journal of Operational Research, 149: 282-301, 2003

[44] V. Valls, S. Quintanilla, F. Ballestin, "**An evolutionary approach to the resource-constrained project scheduling problem**", MIC – 4th Metaheuristics International Conference, 217-220, 2001

[45] L. Vogt, C.A. Poojari, J.E. Beasley, "**A Tabu Search Algorithm for the Single Vehicle Routing Allocation Problem**", Journal of the Operational Research Society, 2006

[46] D. S. Yamashita, "**Scatter Search para Programação de Projetos com Custo de Disponibilidade de Recursos sob Incerteza**", Tese de Doutorado, Unicamp, São Paulo, 2003.