WESB 2013

4th Brazilian Workshop on Search Based Software Engineering

September 29, 2013
Brasília-DF, Brazil

PROCEEDINGS

Volume 01
ISSN: 2178-6097

WESB 2013 CHAIRS

Arilo Claudio Dias Neto
Gledson Elias
Jerffeson Teixeira de Souza
Silvia Regina Vergilio

CBSOFT 2013 GENERAL CHAIRS

Genaína Rodrigues – UnB
Rodrigo Bonifácio – UnB
Edna Dias Canedo - UnB

ORGANIZATION

Universidade de Brasília (UnB)
Departamento de Ciência da Computação (DIMAp/UFRN)

PROMOTION

Brazilian Computing Society (SBC)

SPONSORS

CAPES, CNPq, Google, INES, Ministério da Ciência, Tecnologia e Inovação, Ministério do Planejamento, Orçamento e Gestão e RNP

SUPPORT

Instituto Federal Brasília, Instituto Federal Goiás, Loop Engenharia de Computação, Secretaria de Turismo do GDF, Secretaria de Ciência Tecnologia e Inovação do GDF e Secretaria da Mulher do GDF

ÍNDICE DE ARTIGOS / TABLE OF CONTENTS

PALESTRA CONVIDADA

SESSÃO TÉCNICA 1: PLANEJAMENTO DO PROJETO DE SOFTWARE E REQUISITOS

SESSÃO TÉCNICA 2: TESTE DE SOFTWARE, MAPEAMENTO DA COMUNIDADE DE SBSE, ESTUDOS EXPERIMENTAIS

Uma Abordagem MultiObjetivo para o Problema de Desenvolvimento de Cronogramas de Projetos de Software

Sophia Nóbrega[1], **Sérgio R. de Souza**[1], **Marcone J. F. Souza**[2]

[1]Centro Federal de Educação Tecnológica de Minas Gerais (CEFET/MG)
Av. Amazonas, 7675 – 30.510-000 – Belo Horizonte – MG – Brasil

[2]Departamento de Computação
Universidade Federal de Ouro Preto (UFOP)
Campus Universitário – 35.400-000 – Ouro Preto – MG – Brasil

sophia@dcc.ufmg.br, sergio@dppg.cefetmg.br, marcone@iceb.ufop.br

Abstract. *This paper deals with the Problem of Development Schedules (PDS) for Software Projects. In this study several important aspects are considered, such as availability of resources and skills, project tasks and their interdependencies, deadlines, costs, tasks size estimate, teams productivity and experience. An algorithm based on the multiobjective metaheuristic MOVNS are proposed for solving this problem. The results obtained are competitive and better than those presented by expert project managers.*

Resumo. *Esse artigo trata o Problema de Desenvolvimento de Cronograma (PDC) para projetos de Software. Nesse estudo importantes aspectos são considerados, como disponibilidade dos recursos e suas habilidades, as tarefas do projeto e suas interdependências, deadlines, custos, estimativa de tamanho das tarefas, produtividade das equipes e sua experiência. Para resolver o problema, é proposto o uso de um algoritmo multiobjetivo baseado na metaheurística MOVNS. Os resultados obtidos são competitivos e melhores que aqueles apresentados por experientes gerentes de projetos.*

1. Introdução

Empresas preocupadas em se manterem competitivas buscam sempre reduzir o custo e a duração de seus projetos, mas esses dois objetivos são sempre conflitantes. Existe a necessidade de controlar pessoas e o processo de desenvolvimento, conseguindo alocar eficientemente os recursos disponíveis para executar as tarefas demandadas pelo projeto, sempre satisfazendo uma variedade de restrições.

A abordagem de problemas complexos da Engenharia de Software, como é o caso do problema abordado nesse artigo, utilizando técnicas de otimização, é uma emergente área de pesquisa denominada SBSE *(Search Based Software Engineering)* [Barros and Dias-Neto 2011]. O principal objetivo do SBSE é oferecer mecanismos de apoio ao engenheiro de software para resolver problemas inerentes da Engenharia de Software. Baseado nesses conceitos, a abordagem proposta tem, como objetivo, apoiar e guiar o gerente na atividade de desenvolvimento de cronogramas de projetos de software.

Dentro desse contexto, o presente trabalho apresenta importantes contribuições. É apresentada a primeira abordagem baseada em otimização multiobjetivo usando a

metaheurística MOVNS *(MultiObjective Variable Neighborhood Search)* para resolver o PDC. A segunda contribuição é a proposição de um conjunto de 9 estruturas de vizinhança, com o objetivo de explorar todo o espaço de busca para o problema. Todos os movimentos são descritos na subseção 4.2.

No desenvolvimento desse trabalho, será apresentada uma caracterização detalhada de todos os aspectos e variáveis presentes no dia a dia de um gerente de projeto de software, como habilidades e experiências individuais, custos, prazos, produtividade, horas extras, interdependência das tarefas, duração de tarefas, etc. Para avaliar e validar esta pesquisa, serão realizados estudos empíricos.

2. Trabalhos Relacionados

Em [Antoniol et al. 2005] e [Penta et al. 2011] os autores resolveram, respectivamente, o problema de alocação de pessoas e o problema de cronograma de projetos com alocação de pessoas. Nestes trabalhos, utilizam formulações mono-objetivo e implementam os algoritmos Genéticos, *Simulated Annealing* e *Hill Climbing*. Para validar as implementações, realizaram estudos empíricos com dados de projetos reais, obtendo soluções valiosas para auxiliarem os gerentes de projetos no processo de tomada de decisão. No estudo apresentado por [Penta et al. 2011] também foi modelada uma abordagem multiobjetivo, mas não foi avaliada.

Em [Alba and Chicano 2007], o Problema de Desenvolvimento de Cronograma de projetos é tratado, utilizando-se um Algoritmo Genético clássico, que foi validado utilizando dados fictícios, obtidos a partir de um gerador automático de projetos. Os autores tratam dois objetivos: minimizar o tempo e minimizar o custo do projeto, que são combinados em uma única função objetivo, usando pesos diferentes. Em [Minku et al. 2012], os autores propõem um Algoritmo Evolucionário (AE) multiobjetivo e comparam seu algoritmo com o proposto em [Alba and Chicano 2007].

Em [Gueorguiev et al. 2009], os autores apresentaram a primeira formulação multiobjetivo para esse tipo de problema, no qual robustez e tempo de conclusão do projeto são tratados como dois objetivos concorrentes para resolver o problema de planejamento de projeto de software. O algoritmo SPEA II foi implementado e testado em quatro projetos reais. Os resultados indicam um bom desempenho da abordagem proposta. Em [Colares 2010], o autor utilizou um algoritmo genético multiobjetivo para resolver o problema de alocação de equipes e desenvolvimento de cronogramas. A função de aptidão proposta busca minimizar o tempo total do projeto, o custo total, o atraso nas tarefas e as horas extras.

Segundo o conhecimento dos autores do presente artigo, não foram encontradas referências na literatura ao uso dos algoritmos GRASP e MOVNS para resolver o PDC e, neste sentido, a proposição do uso destas abordagens para este problema como uma contribuição científica de interesse do presente artigo. Observa-se, também, que, na grande maioria dos estudos encontrados na literatura, o PDC é tratado por meio de algoritmos baseados em busca populacional, seja por uma abordagem mono-objetivo ou por uma abordagem multiobjetivo. No presente trabalho, optou-se, por desenvolver algoritmos baseados em busca local para a solução do PDC.

3. Formulação do Problema

A formulação matemática apresentada foi desenvolvida baseada na proposta feita por Colares [Colares 2010]. Algumas características do problema modeladas por Colares foram excluídas, pois não foi possível coletar dados de projetos reais que permitissem o seu uso. O algoritmo proposto recebe, como parâmetros de entrada, tarefas e recursos humanos. As tarefas possuem, como atributos, esforço estimado, nível de importância e datas de inicio e fim. Os recursos humanos são divididos em contratado e empregado. O primeiro possui, como atributos, valor hora e dedicação diária. O empregado possui os atributos salário, dedicação diária, valor hora extra e tempo máximo de hora extra. Ambos os tipos possuem um atributo que representa seu calendário de disponibilidade.

Cada recurso humano possui uma lista de habilidades individuais, com seu respectivo nível de proeficiência, e uma lista de tarefas, com seu respectivo nível de experiência, assim como cada tarefa possui uma lista de habilidades necessárias para sua execução. Cada recurso é alocado a uma determinada tarefa em valores percentuais, que variam de 0 (zero) ao máximo permitido para o recurso.

Para definir a interdependência entre tarefas, ou sequenciamento, a abordagem proposta utiliza os quatro conceitos de relacionamento utilizados pela maioria das ferramentas de gerência de projeto: Início-Início (II), Início-Final (IF), Final-Início (FI) e Final-Final(FF). Além disso, o algoritmo reajusta o início de tarefas que não pertencem ao caminho crítico do projeto, evitando a quebra de restrição de recursos.

A produtividade de um recurso $prod_{r,t}$ pode ser obtida pela fórmula:

$$prod_{r,t} = x_{r,t} r^{dedicacao} \left(\prod_{s \in \left(S^r \bigcap S^t\right)} r^{proef}(s) \right) r^{exp}(t) \tag{1}$$

em que $x_{r,t}$ representa a proporção de esforço do recurso r para executar a tarefa t; $r^{dedicacao}$ é a dedicação diária do recurso r em horas; S^r é o conjunto de habilidades que o recurso r possui; S^t é o conjunto de habilidades requeridos pela tarefa t; $r^{proef}(s)$ é o fator de ajuste devido à proficiência do recurso r na habilidade s; e $r^{exp}(t)$ é o fator de ajuste devido à experiência do recurso r na tarefa t. O tempo de duração de uma tarefa em dias $t^{duracao}$ pode ser obtido pela fórmula:

$$t^{duracao} = \frac{t^{esforco}}{\sum_{r \in R} prod_{r,t}}, \quad \forall t \in T \tag{2}$$

em que $t^{esforco}$ é o esforço da tarefa em Pontos de Função (PF).

Estimado o tempo de duração de cada tarefa $(t^{duracao})$, é calculada a duração total do cronograma do projeto, ou *makespan*. A minimização do *makespan* é o o primeiro objetivo proposto. A duração total do projeto, ou *makespan*, é representada pela função:

$$S = makespan \tag{3}$$

O segundo objetivo proposto é minimizar o custo total do projeto, representado pela função:

$$C = \sum_{i=1}^{R} \sum_{j=1}^{T} (c_{ij} t_j^{duracao}) \tag{4}$$

O custo total é a soma do pagamento dos recursos por sua dedicação no projeto. Esse custo c_{ij} é calculado multiplicando o salário pago por hora para o empregado pelo seu tempo dedicado ao projeto mais as horas extras. O tempo dedicado ao projeto é, então, calculado pela soma da dedicação do recurso multiplicado pela duração de cada tarefa.

4. Modelo Heurístico

4.1. Representação da Solução

Cada solução s do problema é representada por uma matriz bidimensional, denominada Matriz de Alocação X, e um vetor, denominado Vetor Cronograma Y, que armazena o instante de início de cada tarefa.

Uma dimensão da Matriz de Alocação representa os recursos humanos disponíveis $\left\{r_1, r_2, ..., r_{|R|}\right\}$, enquanto a outra dimensão representa as tarefas que devem ser executadas $\left\{t_1, t_2, ..., t_{|T|}\right\}$. Na matriz X, cada variável $x_{r,t}$ recebe um valor inteiro entre 0 (zero) e dedicação diária de cada recurso, acrescida do tempo máximo de hora extra, se for o caso. Esse valor inteiro é interpretado, dividindo-o pela dedicação diária, de forma a se obter porcentagens de 0 a 100%, que representam o esforço dedicado pelo recurso r na execução da tarefa t. Quando o percentual for superior a 100%, indica a realização de hora extra.

O vetor Cronograma possui dimensão T, sendo T o total de tarefas. Os índices do vetor representam as tarefas, e cada posição do vetor é preenchida por um real, que indica o tempo de inicio de execução da tarefa.

4.2. Estruturas de Vizinhança

Para explorar o espaço de soluções foram criados 9 movimentos. Todos os movimentos descritos a seguir são sempre realizados, respeitando-se as restrições de compatibilidade entre recursos e tarefas:

- **Movimento Realocar Recurso entre Tarefas Distintas - $M^{RD}(s)$:** este movimento consiste em selecionar duas células x_{ri} e x_{rk} da matriz X e repassar a dedicação de x_{ri} para x_{rk}. Assim, um recurso r deixa de trabalhar na tarefa i e passa a trabalhar tarefa k.
- **Movimento Realocar Recurso de uma Tarefa - $M^{RT}(s)$:** este movimento consiste em selecionar duas células x_{it} e x_{kt} da matriz X e repassar a dedicação de x_{it} para x_{kt}. Assim, a dedicação de um recurso i é realocada para um recurso k que esteja trabalhando na tarefa t.
- **Movimento Desalocar Recurso de uma Tarefa - $M^{DT}(s)$:** consiste em selecionar uma célula x_{rt} da matriz X e zerar seu conteúdo, isto é, retirar a alocação de um recurso r que estava trabalhando na tarefa t.
- **Movimento Desalocar Recurso no Projeto - $M^{DP}(s)$:** consiste em desalocar toda a dedicação de um recurso r no projeto. O movimento retira todas as alocações do recurso r, que deixa de trabalhar no projeto. O recurso volta a trabalhar no projeto assim que uma nova tarefa for associada a ele.
- **Movimento Dedicação de Recursos - $M^{DR}(s)$:** este movimento consiste em aumentar ou diminuir a dedicação de um determinado recurso r na execução de uma tarefa t. Neste movimento, uma célula x_{rt} da matriz X tem seu valor acrescido ou decrescido em uma unidade.

- **Movimento Troca de Recursos entre Tarefa -** $M^{TB}(s)$**:** duas células x_{ri} e x_{rk} da matriz X são selecionadas e seus valores são permutados, isto é, os recursos que trabalham nas tarefas i e k são trocados.
- **Movimento Troca de Recursos de uma Tarefa -** $M^{TO}(s)$**:** duas células x_{it} e x_{kt} da matriz X são selecionadas e seus valores são permutados, isto é, os recursos i e k que trabalham na tarefa t são trocados.
- **Movimento Insere Tarefa -** $M^{IT}(s)$**:** este movimento consiste em inserir uma tarefa que está em uma posição i em outra posição j do vetor Y. Esse movimento é realizado somente entre tarefas sem relações de precedência.
- **Movimento Troca Tarefa -** $M^{TT}(s)$**:** consiste em trocar duas células distintas y_i e y_k do vetor Y, ou seja, trocar o tempo de inicio de execução das tarefas i e k. Os movimentos de trocas serão feitos sempre respeitando a ordem de precedência para executar as tarefas.

5. Algoritmo Proposto

Nesse artigo, é proposto um algoritmo multiobjetivo, nomeado GRASP-MOVNS, que consiste na combinação dos procedimentos Heurísticos *Greedy Randomized Adaptative Search Procedure* - GRASP [Souza et al. 2010] e *Multiobjective Variable Neighborhood Search* - MOVNS [Geiger 2008]. O algoritmo GRASP-MOVNS foi implementado utilizando a mesma estratégia proposta por [Coelho et al. 2012].

O algoritmo 1 apresenta o pseudocódigo do algoritmo GRASP-MOVNS. Na linha 2, é gerado o conjunto inicial de soluções não dominadas através do procedimento parcialmente guloso *GRASP* [Feo and Resende 1995]. São geradas duas soluções iniciais s_1 e s_2, que são construídas utilizando, como regra de prioridade, as tarefas de maior duração e as tarefas que pertencem ao caminho crítico, respectivamente. A versão do *GRASP* implementada utiliza, como método de busca local, a heurística *Variable Neighborhood Descent* - VND [Hansen and Mladenovic 1997], que envolve a substituição da solução atual pelo resultado da busca local, quando há uma melhora. Porém, a estrutura de vizinhança é trocada de forma determinística, cada vez que se encontra um mínimo local. A solução resultante é um mínimo local em relação a todas as nove estruturas de vizinhanças: M^{IT}, M^{TT}, M^{DR}, M^{RD}, M^{RT}, M^{DP}, M^{DT}, M^{TB}, M^{TO}.

Nas linhas 6 e 7 do Algoritmo 1 é feita a seleção de um indivíduo do conjunto de soluções não-dominadas D, marcando-o como "visitado". Quando todos os indivíduos estiverem marcados como visitados, a linha 32 retira estes marcadores. As variáveis $level$ e $shaking$, mostradas nas linhas 3 e 4 do Algoritmo 1, regulam a perturbação utilizada no algoritmo. Esta versão do algoritmo MOVNS, proposta por [Coelho et al. 2012], possui um mecanismo que regula o nível de perturbação do algoritmo, ou seja, a variável *shaking* é incrementada quando o algoritmo passa um determinado tempo sem obter boas soluções. Da linha 9 à 12 do Algoritmo 1 ocorre o laço de perturbação do algoritmo. A heurística AdicionarSolucao, mostrada no Algoritmo 2, é acionada na linha 16, e adiciona, ao conjunto solução D, as soluções geradas pelo GRASP-MOVNS. No mecanismo utilizado, quanto maior o valor da variável *shaking*, maior será a intensidade da perturbação na solução. Para cada unidade dessa variável, aplica-se um movimento aleatório, dentre as nove vizinhanças: M^{IT}, M^{TT}, M^{DR}, M^{RD}, M^{RT}, M^{DP}, M^{DT}, M^{TB}, M^{TO}. A variável *level* regula quando a variável *shaking* será incrementada. As linhas 24 e 25 retornam os

Algoritmo 1: GRASP-MOVNS

Entrada: *Vizinhança $N_K(s)$; graspMax; levelMax*
Saída: *Aproximação de um conjunto eficiênte D*

início
  $D \leftarrow GRASP(graspMax)$
  $level \leftarrow 1$
  $shaking \leftarrow 1$
  enquanto (*Critério de parada não satisfeito*) **faça**
    Seleciona uma solução não visitada $s \in D$
    Marque s como visitada
    $s \leftarrow s'$
    para $i \rightarrow shaking$ **faça**
      Selecione aleatóriamente uma vizinhança $N_k(.)$
      $s' \leftarrow$ *Pertubação(s', k)*
    fim
    $k_{ult} \leftarrow k$
    $incrementa \leftarrow verdadeiro$
    para $s'' \in N_{k_{ult}}(s')$ **faça**
      $adicionarSolucao$(D, s", f(s"), added)
      se $adicionado = verdadeiro$ **então**
        $incrementa \leftarrow falso$
      fim
    fim
    se $incrementa = verdadeiro$ **então**
      $level \leftarrow level + 1$
    senão
      $level \leftarrow 1$
      $shaking \leftarrow 1$
    fim
    se $level \geq levelMax$ **então**
      $level \leftarrow 1$
      $shaking \leftarrow shaking + 1$
    fim
    se *todo $s \in D$ estão marcadas como visitadas* **então**
      Marque todos $s \in D$ como não visitado
    fim
  fim
  Retorne D
fim

valores das variáveis *level* e *shaking* para uma unidade, quando pelo menos uma solução é adicionada ao conjunto eficiente D.

Algoritmo 2: adicionarSolucao

Entrada: conjunto D, s',$z(s')$
Saída: $conjuntoD, adicionado$

início
  $adicionado \leftarrow verdadeiro$
  para $s \in D$ **faça**
    se $z(s) \leq z(s')$ **então**
      $adicionado \leftarrow falso$
      $break$
    fim
    se $z(s') < z(s)$ **então**
      $D = D\ s$
    fim
  fim
  se $adicionado = verdadeiro$ **então**
    $D = D \bigcup s'$
  fim
  Retorne D, adicionado
fim

5.1. Experimentos e Análise

Os algoritmos foram implementados em linguagem java, e os experimentos realizados em um notebook Dell Inspirion 14 Core i3-3110M, 3 MB Cache, 2.4 GHz, 4GB de RAM, sob sistema operacional windows 7. Para realizar os testes, foi utilizado um conjunto de 3 instâncias, disponível em `www.decom.ufop.br/prof/marcone/projects/SBSE.html`. Os dados das instâncias são de projetos reais de duas empresas de desenvolvimento de sistemas de Belo Horizonte, que estão no mercado de há mais de 8 anos.

Foram criados 3 cenário de teste. O cenário A de teste possui 10 casos de uso, totalizando 72 tarefas, 21 recursos e 7 habilidades envolvidas. O cenário B possui 16 casos de uso totalizando 100 tarefas, 16 recursos e 4 habilidades. O cenário C possui 20 casos de uso com 120 tarefas, 11 recursos e 4 habilidades. Todos os recursos possuem uma carga horária de 8 horas diárias e com salários de acordo com a função desempenhada e experiência. O algoritmo foi configurado durante os testes para não permitir que sejam feitas horas extras. Optou-se por, inicialmente, não permitir que sejam realizadas horas extras, pois esta é uma prática comum na indústria de desenvolvimento de software. Em geral, durante o desenvolvimento inicial de um cronograma de projeto, não são consideras horas extras, que são permitidas apenas em casos pontuais no dia a dia dos projetos. Cada recurso possui suas próprias habilidades e cada tarefa possui uma lista de habilidades requeridas para sua execução. O limite máximo e mínimo de recursos alocados foi registrado com base nas necessidades de cada tarefa executada. Sempre é alocado pelo menos um recurso que possua cada uma das habilidades requeridas para execução da tarefa. Como os dados utilizados não especificavam proficiência e experiência dos recursos, foi considerado sempre o valor "normal", em vista desses fatores não influenciarem no cálculo de produtividade das equipes.

Os dados dos projetos de teste foram exportados do MS Project [Microsoft 2011] para um arquivo texto, seguindo uma sintaxe pré-definida. No início da execução do algoritmo GRASP-MOVNS, o arquivo texto é importado de forma automatizada, inicializando os objetos modelados para o PDC. Após inicializar os dados do projeto, são carregados os parâmetros dos algoritmos desenvolvidos. Após a realização de alguns testes iniciais, foram definidos os seguintes valores para os parâmetros: $graspMax = 150$ e $levelMax = 10$. A variável $shaking$ teve seu valor máximo fixado em 7, pois, durante a fase de construção, foram utilizadas apenas 7 movimentos dentre os 9 propostos. Os movimentos "Realocar Recurso de uma Tarefa"e "Troca de Recursos de uma Tarefa"foram desconsiderados na fase de construção, pois, durante os testes, observou-se que geram uma perturbação pequena nas soluções. Os valores das variáveis $shaking$ e $levalMax$ foram definidos altos, para permitir uma maior intensidade da perturbação na solução. O valor de $graspMax$ indica o número de iterações executadas na fase de construção GRASP. Como critério de parada do algoritmo GRASP-MOVNS, foi utilizando um tempo de execução igual a 180 segundos.

Para avaliar o algoritmo proposto nesse artigo, foi criado um grupo formado por 3 experientes gerentes de projeto de software. Todos os indivíduos participantes receberam os cenários de testes e, utilizando somente seu conhecimento e experiência na atividade de gerência de projetos, construíram o cronograma do projeto. Para realizar a tarefa, os indivíduos dedicaram 4 horas para construir o cronograma e utilizaram o software MS Project 2010. As soluções obtidas foram coletadas e confrontadas com resultados do al-

goritmo, para validar a competitividade e a qualidade das soluções geradas pelo algoritmo GRASP-MOVNS. A próxima subseção apresenta e analisa os resultados obtidos.

5.2. Análise dos Resultados

As Figuras 1, 2 e 3 apresentam os gráficos confrontando os resultados dos gerentes com os obtidos em 30 execuções de cada cenário de teste. Pode-se afirmar que a abordagem proposta nesse artigo produziu soluções comparáveis as dos gerentes, inclusive melhores e mais diversificadas no aspecto geral. Em geral, o algoritmo GRASP-MOVNS produziu melhores soluções na minimização dos dois objetivos propostos, tempo e custo.

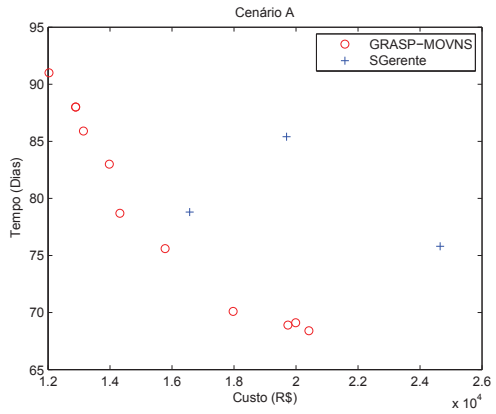

Figura 1. Comparação entre o algoritmo GRASP-MOVNS e as soluções do Gerentes.

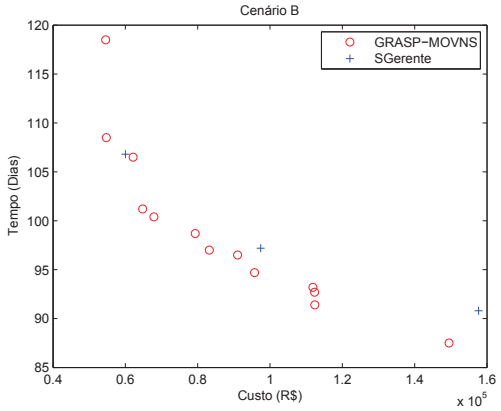

Figura 2. Comparação entre o algoritmo GRASP-MOVNS e as soluções do Gerentes.

Em relação à aplicabilidade da abordagem em projetos de desenvolvimento de software, principal objetivo de análise destes cenários de teste, tem-se como atingido o resultado esperado. Com sucesso, projetos reais de desenvolvimento de software foram adaptados à abordagem, obtendo-se soluções para o planejamento de diversas tarefas do projeto em relação à alocação de equipes e ao desenvolvimento de cronograma.

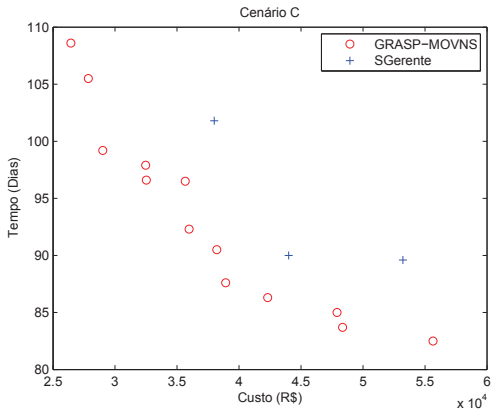

Figura 3. Comparação entre o algoritmo GRASP-MOVNS e as soluções do Gerentes.

6. Conclusões

Este trabalho propôs a utilização de uma abordagem multiobjetivo com o objetivo de encontrar a melhor alocação e o melhor sequenciamento das atividades do projeto, a fim de elaborar bons cronogramas que minimizem o tempo e o custo do projeto. O algoritmo GRASP-MOVNS, proposto nesse artigo para resolver o PDC, obteve resultados competitivos, apresentando resultados melhores que os encontrados por experientes gerentes de projetos.

O Desenvolvimento do Cronograma do Projeto de Software é um problema de difícil resolução, devido à grande quantidade de restrições envolvidas. As principais restrições são implementadas, como alocações em que mais de um recurso pode ser alocado por tarefa e um recurso pode trabalhar em mais de uma tarefa no mesmo dia, disponibilidade de recursos, carga horária e salário individual, quatro tipos de ligação de dependência entre tarefas, e tipos de tarefas. No entanto, existem muitas outras características que podem ficar para trabalhos futuros, como *overhead* de comunicação, coletar dados de projetos que permitam o uso dos coeficientes de experiência e proficiência dos recursos, a permissão de realização de horas extras e a avaliação do impacto dessa configuração para as soluções geradas pela abordagem.

Com o objetivo de melhorar a qualidade das soluções, podem ser feitos aprimoramentos no algoritmo, como a utilização de outras técnicas de otimização e a criação de novas heurísticas. Também pode ser melhor investigada a eficiência e eficácia do algoritmo proposto em relação a outras técnicas de otimização, em especial a técnicas que utilizem algoritmos genéticos. Finalmente, realizar novos estudos de caso, aplicando a abordagem em outros projetos reais com o objetivo de analisar o comportamento da abordagem em diferentes cenários.

Agradecimentos

Os autores agradecem a FAPEMIG, CNPq e CEFET-MG por apoiar o desenvolvimento dessa pesquisa.

Referências

Alba, E. and Chicano, F. (2007). Software project management with GAs. *Information Sciences*, 177(11):2380–2401.

Antoniol, G., Penta, M. D., and Harman, M. (2005). Search-based techniques applied to optimization of project planning for a massive maintenance project. *In Proceedings of the 21st IEEE International Conference on Software Maintenance*, pages 240–249.

Barros, M. O. and Dias-Neto, A. C. (2011). Desenvolvendo uma abordagem sistemática para avaliação dos estudos e xperimentais em Search Based Software Engineering. *II Workshop de Engenharia de Software Baseada em Buscas - WESB*, 12:49–56.

Coelho, V. N., Souza, M. J. F., Coelho, I. M., Guimarães, F. G., and Lust, T. (2012). Algoritmos multiobjetivos para o problema de planejamento operacional de lavra. In *Anais do XV Simpósio de Pesquisa Operacional e Logística da Marinha (SPOLM 2012)*.

Colares, F. (2010). Alocação de equipes e desenvolvimento de cronogramas em projetos de software utilizando otimização. Dissertação de mestrado, UFMG.

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

Geiger, M. J. (2008). Randomized variable neighborhood search for multi objective optimization. *Proceedings of the 4th EU/ME Workshop: Design and Evaluation of Advanced Hybrid Meta-Heuristics*, pages 34–42.

Gueorguiev, S., Harman, M., and Antoniol, G. (2009). Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering. *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (CECOO)*, pages 1673–1680.

Hansen, P. and Mladenovic, N. (1997). Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100.

Microsoft (2011). MS Project 2010. [Online; accessed 10-December-2011].

Minku, L. L., Sudholt, D., and Yao, X. (2012). Evolutionary algorithms for the project scheduling problem: runtime analysis and improved design. In *GECCO'12*, pages 1221–1228.

Penta, M. D., Harman, M., and Antoniol, G. (2011). The use of search-based optmization techniques to schedule and staff software projects: an approach and an empirical study. *Software - Practice and Experience*, 41:495–519.

Souza, M. J. F., Coelho, I. M., Ribas, S., Santos, H. G., and Merschmann, L. H. C. (2010). A hibrid heuristic algorithm for the open-pit-mining operacional planning problem. *European Journal of Operational Research*, 207(2):1041–1051.