Resumo Expandido

Título da Pesquisa (Português): Desenvolvimento de um *Software* de Apoio ao Planejamento de Projetos (ProScenario) - Algoritmos		
Título da Pesquisa (Inglês): Development of a software to Project Planning (ProScenario) - Algorithms		
Palavras-chave: restrição de recursos, planejamento de projetos, escalonamento de projetos, RCPSP, heurísticas		
Keywords: resource constraint, Project planning, Project scheduling, RCPSP, heuristics		
Campus: Formiga	**Tipo de Bolsa**: PIBIT/CNPq	**Financiador**: CNPq
Bolsista(s): Natanael Ramos		
Professor Orientador: Diego Mello da Silva		
Área de Conhecimento: 3.08.08.00-8: Pesquisa Operacional, 1.03.03.02-2: Engenharia de *Software*, 1.03.02.02-6: Modelos Analíticos e de Simulação		**Edital**: 139/2013

Resumo: Este trabalho apresenta os resultados do desenvolvimento do módulo de decisão do software ProScenario até o momento. Tal módulo de decisão implementa algoritmos para lidar com o Problema de Escalonamento de Projetos com Restrição de Recursos, problema de otimização combinatória conhecido como RCPSP (*Resource Constraint Project Scheduling Problem*). Ele provê interface de integração com os demais módulos do ProScenario no intuito de facilitar a tomada de decisão em planejamento de projetos com restrições de recursos. Os algoritmos implementados até o momento foram os da técnica PERT/CPM, o construtor de cronogramas SGS Serial e heurísticas construtivas para sequenciamento de atividades restritas. O próximo passo é a implementação das metaheurísticas GRASP e *Simulated Annealing*, ambas já projetadas e avaliadas previamente em outro trabalho, além do módulo de simulação usando Monte Carlo.

Abstract: This work presents the results of the development of the decision module of the software Proscenario. This decision module implements algorithms to manage projects with resource constraints, a combinatorial problem known as RCPSP (Constraint Project Resource Scheduling Problem). It provides an integration interface with other ProScenario modules in order to aid decision making in planning projects with resource constraints. In this work we have implemented the follow algorithms: PERT/CPM Technique, SGS Serial and constructive heuristics for constraint activity scheduling. The next step consists on the implementation of the GRASP and Simulated Annealing metaheurists, both previously designed and evaluated in another work and Monte Carlo simulation to risk analysis on project costs and time.

INTRODUÇÃO

Um projeto consiste em uma série de tarefas ou atividades que têm um objetivo específico a ser completado sob certas especificações, com datas de início e fim definidas, limites de financiamento, e consumindo recursos para sua execução (Kerzner, 2001). Segundo Guido e Clements (2009), o objetivo de qualquer projeto é completar o escopo do trabalho dentro do orçamento em tempo para a satisfação do cliente. Para atingir tal objetivo, é importante desenvolver um plano antes do início do projeto, que deve incluir todas as tarefas a serem executadas, custos associados e estimativas de tempo necessário para completá-las. A falta de planejamento aumenta o risco de não cumprimento do escopo completo do projeto dentro do orçamento e do cronograma de execução do projeto. Para cumprir esta tarefa faz-se necessário o uso de softwares capazes de planejar projetos considerando inclusive restrições de força de trabalho (recursos).

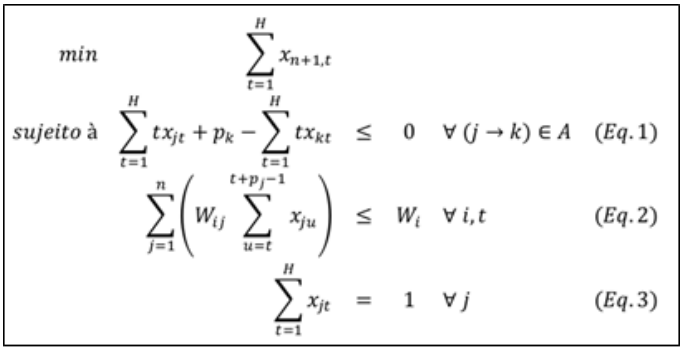
Este trabalho tem por objetivo apresentar os resultados obtidos com a implementação do módulo de tomada de decisão do *software* ProScenario. Tal projeto tem como objetivo principal o desenvolvimento de um *software* de apoio aos gerentes nas tomadas de decisão que envolvam geração de cronogramas de projetos com restrição de recursos, utilizando técnicas baseadas em heurísticas construtivas e metaheurísticas. Neste trabalho serão apresentados os resultados apenas do módulo de decisão do *software* ProScenario. Discussões sobre os demais módulos, modelagem e detalhes de implementação do ProScenario estão fora do escopo.

METODOLOGIA

O projeto de pesquisa iniciou com uma revisão bibliográfica sobre o tema de gerenciamento de projetos, onde estudou-se técnicas e ferramentas tais como: representação hierárquica das fases de um projeto por estrutura analítica de projetos (EAP/WBS) (WYSOCKI; MCGARY, 2003), estimativa de duração de atividades usando estimativa de três pontos e técnica Delphi (WYSOCKI; MCGARY, 2003), sequenciamento de atividades de um projeto usando PERT/CPM (SWANSON; WOOLSEY, 1974), representação de um cronograma de atividades usando diagramas de redes e gráficos de Gantt (WYSOCKI; MCGARY, 2003). O conteúdo estudado foi baseado em metodologias tradicionais de gerência de projetos. Muitas obras tratam do assunto mas não exploram o planejamento de projetos com restrição de recursos, dando ênfase apenas no planejamento e escalonamento de projetos que respeitam apenas restrições de precedência entre atividades.

O *software* ProScenario lida com planejamento de projetos restritos. Complementou-se o estudo aprofundando no problema de otimização combinatória Escalonamento de Projetos com Restrição de Recursos (RCPSP – *Resource Constraint Project Scheduling Problem*) (HARTMANN, 1999; KOLISCH, 1995).

Pinedo (2008) apresenta uma formulação inteira para resolver o RCPSP que é capaz de minimizar a duração total do projeto. No modelo, n atividades contidas em um conjunto A de atividades devem ser escalonadas no tempo. O tempo necessário para executar a j-ésima tarefa é dado por p_j. Uma atividade *dummy* $(n+1)$ é introduzida com tempo de execução $p_{n+1} = 0$, que sucede todas as demais atividades. Seja x_{jt} uma variável binária que assume valor 1 se a atividade j é completada exatamente no tempo t, ou 0 caso contrário. O número total de recursos do tipo i disponíveis para o projeto é dado por W_i. O número de recursos do tipo i que a atividade j demanda durante o intervalo de tempo $[t-1, t]$ é dado por $W_{ij} = \sum_{u=t}^{t+p_j-1} x_{ju}$. Seja $H = \sum_{j=1}^{n} p_j$ um limite superior para a duração total do projeto. Logo, o tempo necessário para completar a atividade j é expresso por $\sum_{t=1}^{H} t x_{jt}$, e a duração total do projeto é dada por $\sum_{t=1}^{H} x_{n+1,t}$. O modelo de programação inteira para o problema do escalonamento de projetos com restrição de recursos é apresentado na Figura 1.



$$min \quad \sum_{t=1}^{H} x_{n+1,t}$$

$$sujeito\ à \quad \sum_{t=1}^{H} t x_{jt} + p_k - \sum_{t=1}^{H} t x_{kt} \leq 0 \quad \forall\, (j \rightarrow k) \in A \qquad (Eq.1)$$

$$\sum_{j=1}^{n} \left(W_{ij} \sum_{u=t}^{t+p_j-1} x_{ju} \right) \leq W_i \quad \forall\, i,t \qquad (Eq.2)$$

$$\sum_{t=1}^{H} x_{jt} = 1 \quad \forall\, j \qquad (Eq.3)$$

Figura 1 - Modelo matemático do problema RCPSP

Embora o modelo seja capaz de expressar uma solução exata para o problema, na prática o problema RCPSP é conhecido ser da classe NP-Difícil (HARTMANN, 1999; KOLISCH, 1995), o que torna improvável existir algum algoritmo determinístico eficiente para resolvê-lo. Estudou-se abordagens heurísticas que embora não sejam capazes de encontrar soluções ótimas para o RCPSP, podem retornar soluções de boa qualidade em tempo polinomial. Nesta seção apresentaremos algumas heurísticas construtivas já implementadas no projeto, um gerador de cronogramas baseado em lista de prioridades, e propostas de heurísticas GRASP (FEO; RESENDE, 1995) e *Simulated Annealing* (KIRKPATRICK, 1983) para construir cronogramas restritos.

Uma das técnicas implementadas foi a combinação de heurísticas construtivas com esquemas geradores de cronograma (SGS – *Schedule Generation Schemes*). As heurísticas construtivas utilizam informações oriundas do método PERT, do diagrama de rede, dos recursos requeridos por atividades e outros, e serão descritas brevemente na Tabela 1. A coluna da esquerda apresenta o nome da heurística e a da direita o critério usado para ordenar as atividades do projeto. Em todos os casos as heurísticas construtivas descritas respeitam a restrição de precedência de atividades, isto é, uma atividade não pode ser escalonada antes de suas predecessoras.

Tabela 1 - Regras de Prioridade

REGRAS DE PRIORIDADE	
Baseadas em Informações de Atividades	
Shortest Processing Time (SPT)	Ordem crescente das durações das atividades.
Longest Processing Time (LPT)	Ordem decrescente das durações das atividades.
Baseadas na Rede de Atividades	
Most Immediate Successors (MIS)	Ordem decrescente de acordo com a quantidade de sucessores imediatos.
Most Total Successors (MTS)	Ordem decrescente de acordo com a quantidade de sucessores imediatos e não-imediatos.
Least Non-Related Jobs (LNRJ)	Uma atividade (*Job*) é relacionada a outra atividade se existe precedência direta ou indireta entre elas. As atividades são justapostas em ordem crescente da quantidade de atividades que não são relacionadas.
Greatest Rank Positional Weight (GRPW)	Ordem crescente de acordo com a soma da duração da atividade somada às durações das atividades sucessoras imediatas.
Baseadas nas informações de escalonamento	
Earliest Start Time (EST)	Ordem crescente de acordo com o tempo de início mais adiantado.

Earliest Finish Time (EFT)	Ordem crescente de acordo com o tempo de término mais adiantado.
Latest Start Time (LST)	Ordem crescente de acordo com o tempo de início mais tardio.
Latest Finish Time (LFT)	Ordem crescente de acordo com o tempo de término mais tardio.
Minimum Slack (MSLK)	Ordem crescente de acordo com o tempo de folga da atividade.
Baseadas em informações dos recursos[1]	
Greatest Resource Work Content (GRWC)	Coloca as atividades em ordem decrescente de acordo com o valor do RWC.
Greatest Cumulative Resource Work Content (GCRWC)	Coloca as atividades em ordem decrescente de acordo com o valor do somatório do RWC da atividade e de cada atividade sucessora imediata.

O algoritmo SGS opera em conjunto com as heurísticas construtivas, recebendo como entrada uma lista de atividades ordenadas segundo o critério da heurística. De posse desta lista o SGS constrói um cronograma restrito. O SGS pode ser encontrado em duas variações: serial e paralelo (HARTMANN, 1999; KOLISCH, 1995). Para o projeto em questão, optou-se pela implementação da abordagem serial. As outras técnicas de solução estudadas e que serão empregadas neste projeto são as metaheurísticas *Simulated Annealing* e GRASP adaptadas para o RCPSP. Ambas já foram estudadas em um trabalho PIBIC anterior (titulo: “Análise Experimental De Metaheurísticas Para O Problema De Escalonamento De Projetos Com Restrição De Recursos”), tendo sido validadas em projeto experimental, e geram bons resultados com tempo de execução hábil.

Ambos o software ProScenario e seu módulo de decisão foram implementados na linguagem C Sharp que utiliza o paradigma de Programação Orientada à Objetos (OOP). Estudou-se suas principais características assim como a ferramenta Microsoft Visual Studio. De posse destes conhecimentos iniciou-se a implementação do módulo de tomada de decisão que é integrado com o software ProScenario na sua camada de negócios, e cuja invocação é feita mediante interface de serviços (SOA).

Para a análise de risco sobre o custo e prazo dos cronogramas gerados será utilizado a simulação de Monte Carlo. Estudou-se os fundamentos da técnica, assim como as distribuições de probabilidade mais conhecidas para simulação. Implementou-se classes que geram números aleatórios com viés de certas distribuições probabilísticas para representar a duração das atividades do projeto. A escolhida segue a distribuição de três pontos conhecida por Beta-PERT, construída a partir de três estimativas: otimista, mais provável e pessimista. O próximo passo é a implementação das metaheurísticas GRASP e *Simulated Annealing* e do módulo de simulação para análise de risco. O ProScenario e módulo de decisão serão hospedados em um servidor Dell PowerEdge T420 com processadores Intel Xeon E5-2430, 2.5 GHz, 15 Mb de cache, com 6 núcleos/12 threads, 12 Gb, disco rígido de 1Tb SATA. A presença de processadores multi-core darão suporte a simulação de cenários via múltiplas *threads* de execução de forma a reduzir o tempo de resposta. A próxima seção apresentará os resultados obtidos até o presente momento sob o ponto de vista dos algoritmos implementados ou projetados.

RESULTADOS E DISCUSSÕES:

[1] *Resource Work Content* (RWC)*:* produto da duração da atividade pela quantidade de recursos requeridos.

Esta seção descreverá os principais algoritmos do módulo de decisão. Atualmente o módulo possui 36 classes implementadas em 3949 linhas de código. Serão descritos brevemente os algoritmos PERT/CPM, o gerador de cronogramas SGS Serial e as metaheurísticas GRASP e Simulated Anneling para o RCPSP. Os pseudocódigos apresentados nessa seção representam as operações em alto nível realizadas pelos métodos PERT/CPM (Algoritmo 1), SGS serial (Algoritmo 2), GRASP (Algoritmo 3), *Simulated Annealing* (Algoritmo 4). Detalhes serão comentados nos próximos parágrafos.

O algoritmo PERT/CPM (Algoritmo 1) foi baseando em Swanson e Woolsey (1974). Como resultado do PERT cada atividade da rede é rotulada com os seguintes valores: ES (*Early Start*), ou tempo de início mais breve; EF *(Early Finish*), ou tempo de término mais breve; LS (*Late Start*) ou tempo de início mais tardio; e LF (*Late Finish*) ou tempo de término mais tardio. Basicamente é constituído de dois procedimentos principais, o *forward-pass*, que calcula o ES e EF, e o *backward-pass*, que calcula o LS e LF. A diferença LS-LF ou ES-EF é denominada folga; atividades com folga igual a zero fazem parte do caminho crítico (não podem atrasar).

```
1: function FORWARD-PASS(rede, atividades-pert)
2:     fila ← atividades-pert
3:     while (fila ≠ ∅) do
4:         atual ← ATIVIDADE-CANDIDATA(atividades-pert)
5:         fila.REMOVE(atividades-pert[atual])
6:         for (s in atual.SUCESSORAS()) do
7:             atividades-pert[s].ES ← MAIOR-EF-PREDECESSORAS(s)
8:             duracao ← s.duracao
9:             atividades-pert[s].EF ← atividades-pert[s].ES + duracao
10:            s.visitada ← TRUE
11:        end for
12:    end while
13: end function

14: function BACKWARD-PASS(rede, atividades-pert)
15:    fila ← atividades-pert
16:    while (fila ≠ ∅) do
17:        atual ← ATIVIDADE-CANDIDATA(atividades-pert)
18:        fila.REMOVE(atividades-pert[atual])
19:        for (p in atual.PREDECESSORAS()) do
20:            atividades-pert[p].LF ← MENOR-LS-SUCESSORAS(p)
21:            duracao ← p.duracao
22:            atividades-pert[p].LS ← atividades-pert[p].LF − duracao
23:            p.visitada ← TRUE
24:        end for
25:    end while
26: end function
```

Algoritmo 1 - PERT/CPM

```
1: function SGS-SERIAL(atividades, projeto)
2:     makespan ← CALCULA-PIOR-MAKESPAN(projeto)
3:     for (a in atividades) do
4:         escalonavel ← TRUE
5:         duracao ← a.duracao
6:         pti ← 0
7:         pti ← MAIOR-TEMPO-TERMINO-PREDECESSORAS(a)
8:         for (i ← pti to makespan) do
9:             for (j ← 1 to a.recursos) do
10:                sup ← i+duracao
11:                if (TEM-RECURSO-DISPONIVEL(j,i,sup, qtd)==FALSE)
   then
12:                    escalonavel ← FALSE
13:                    break
14:                end if
15:            end for
16:            if (escalonavel==TRUE) then
17:                a.inicio ← i
18:                a.termino ← i+duracao
19:                break
20:            else
21:                escalonavel ← TRUE
22:            end if
23:        end for
24:    end for
25: end function
```

Algoritmo 2 - SGS serial

```
1: function SOLUCAO-FASE-CONSTRUTIVA( )
2:     candidatos ← GERA-SOLUCOES( )
3:     rcl ← SELECIONA-SOLUCOES( )
4:     index ← RAND( ) mod rcl.tamanho
5:     return rcl[index]
6: end function

7: function BUSCA-LOCAL(solucao)
8:     atual ← solucao
9:     while (TEM-VIZINHO-MELHOR(atual)) do
10:        atual ← MELHOR-VIZINHO(atual)
11:    end while
12:    solucao ← atual
13:    return solucao
14: end function

15: function GRASP(MaxIter)
16:    for (i ← 0 to MaxIter) do
17:        solucao ← SOLUCAO-FASE-CONSTRUTIVA( )
18:        vizinha ← BUSCA-LOCAL(solucao)
19:        if (vizinha.makespan < solucao.makespan) then
20:            solucao ← vizinha
21:        end if
22:    end for
23: end function
```

Algoritmo 3 – GRASP e seus procedimentos

```
1: function SIMULATED-ANNEALING(atividades-pert, projeto, numPassos, maxI-
   ter)
2:     s ← CONSTROI-SOLUCAO-INICIAL( )
3:     for i ← 0 to maxIter do
4:         temperatura ← CALCULA-TEMPERATURA-INICIAL( )
5:         s* ← s
6:         for (j ← 0 to numPassos) do
7:             nVizinhos ← CALCULA-NUMERO-VIZINHOS( )
8:             for (k ← 0 to nVizinhos) do
9:                 viz ← GERA-VIZINHO(s)
10:                δ ← viz.makespan - s.makespan
11:                if (δ < 0) then
12:                    ATUALIZA-SOLUCOES(s*, s, viz)
13:                else
14:                    if (CRITERIO-ACEITACAO(temperatura)) then
15:                        ATUALIZA-SOLUCOES(s, viz)
16:                    end if
17:                end if
18:            end for
19:            REDUZ-TEMPERATURA(temperatura)
20:        end for
21:    end for
22: end function
```

Algoritmo 4 - *Simulated Annealing*

O algoritmo SGS (Algoritmo 2) serial implementado foi baseado em (HARTMANN, 1999) e opera recebendo uma lista de atividades ordenadas segundo alguma prioridade, construídas pelas heurísticas construtivas, GRASP ou *Simulated Annealing*. O algoritmo busca encaixar cada atividade do cronograma respeitando a ordem da lista de entrada, iniciando cada atividade o mais breve possível respeitando (i) a restrição de precedência e (ii) restrição de recursos disponíveis. Na inexistência de recursos disponíveis no tempo mais breve, o início da atividade é postergada. Escalona-se apenas uma atividade por vez, sempre respeitando a ambas as restrições.

A metaheurística *Simulated Annealing* (Algoritmo 4) baseia-se no trabalho de Bouleimen e Lecocq (2003) que parte de uma solução inicial construída pela heurística SPT (*Shortest Processing Time*) e realiza busca local simulando o procedimento termodinâmico de recozimento. A estrutura de vizinhança é construída pela perturbação da lista de atividades ordenada segundo a técnica *Shift Cycling* descrita pelos autores. A metaheurística GRASP (Algoritmo 3) projetada opera em duas fases. Na primeira, constrói-se uma lista de atividades utilizado variantes das heurísticas construtivas apresentadas na Tabela 1. De posse dessa lista, aplica-se a fase de busca local operando de maneira semelhante ao algoritmo *Simuated Annealing* de Bouleimen e Lecocq (2003). Em ambos os casos as heurísticas mostraram-se ágeis para construir cronogramas restritos sendo consideradas adequadas para a simulação de diversos cenários que variam a disponibilidade de recursos do projeto. Tais cenários oferecerão um leque de opções com diferentes prazos e custos para o gerente de projetos, que poderá optar pela melhor escolha dentro de suas limitações.

CONCLUSÕES

Os algoritmos implementados até o momento tiveram sua corretude validada por meio de ensaios experimentais simples, já encontram-se integrados com as classes pertencentes à camada de regra de

negócios do *software* ProScenario. Os próximos passos deste projeto são a implementação das metaheurísticas GRASP e Simulated Anneling e da simulação de Monte Carlo. Ao finalizar o módulo estará pronto para operar completamente com o *software* ProScenario, que será disponibilizado para a comunidade de gerentes de projetos. Espera-se com isso preparar melhor os gerentes e equipes de projeto para lidar com planejamento de demandas, serviços e produtos cuja execução é limitada por escassez de força de trabalho.

REFERÊNCIA BIBLIOGRÁFICA

BOULEIMEN, K.; LECOCQ, H. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. **European Journal of Operational Research**, v. 149, n. 2, p. 268-281, 2003.

GUIDO J., CLEMENTS J. P. (2009) **Successfull Project Management**. 4 ed. Cengage Learning, ISBN: 0324656130.

HARTMANN, S. **Project Scheduling under Limited Resources: Models, Methods and Applications**. Springer-Verlag Berlin Heidelberg GmbH, 1999.

Kerzner, H. (2001). **Project Management - A Systems Approach to Planning, Scheduling, and Controlling.** 7 ed. John Wiley and Sons.

KIRKPATRICK, Scott et al. Optimization by simulated annealing. **science**, v. 220, n. 4598, p. 671-680, 1983.

KOLISCH, R. **Project Scheduling under Resource Constraints**. Springer-Verlag Berlin Heidelberg GmbH, 1995.

FEO, T. A.; RESENDE, M. G. C. Greedy Randomized Adaptive Search Procedures. **Journal of Global Optimization**, [S.I.], 1995. n. 2, p. 109-133.

PINEDO, M. L. (2008). Scheduling: Theory, Algorithms and Systems (3rd Edition). Springer.

SWANSON, H., & WOOLSEY, R. **A PERT-CPM TUTORIAL**. ACM SIGMAP Bulletin, v. 16, 1974. 54-62 p.

WYSOCKI, R. K., & MCGARY, R. (2003). Effective Project Management : Traditional, Adaptative, Extreme. 3. ed. Wiley Publishing, Inc.