risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Revista Ibérica de Sistemas y Tecnologías de Información

Recebido / Recibido: 24/10/2010
Aceitação / Aceptación: 03/12/2010

Um Algoritmo Genético para Programação de Projectos em Redes de Actividades com Complementaridade de Recursos

Helder C. Silva [1], José A. Oliveira [2], Anabela P. Tereso [2]

helder@ifam.edu.br, zan@dps.uminho.pt, anabelat@dps.uminho.pt

[1] IFAM – Instituto Federal de Educação Tecnológica do Amazonas, Manaus, Brazil

[2] University of Minho, Guimarães, Portugal

Resumo: Neste artigo abordamos a questão da alocação óptima dos recursos, mais especificamente, a análise da complementaridade dos recursos (recurso principal ou recurso-*P* e do recurso de suporte ou recurso-*S*) às actividades de um projecto. Neste artigo apresenta-se um algoritmo genético, baseado num alfabeto de chaves aleatórias, e analisa-se o procedimento para criar soluções a partir de um cromossoma.

Palavras-chave: Gestão de Projectos, Programação de Projectos, Complementaridade de Recursos.

Abstract: We address the issue of optimal resource allocation, and more specifically, the analysis of complementarity of resources (primary resource or P-resource and supportive resource or S-resource) to activities in a project. In this paper we present a Genetic Algorithm, based in a random keys alphabet, and we analyse the procedure to build solutions from a chromosome.

Keywords: Project Management, Scheduling, Complementarity of resources.

1. Introdução

A Gestão de Projectos é actualmente uma actividade importante em vários sectores da sociedade pós-industrial que inclui agora também a produção de informação. Torna-se fulcral ter-se uma clara percepção das diferentes fases do ciclo de vida de um projecto, dos processos, técnicas e ferramentas, adequados à gestão de um projecto, tendo em atenção a especificidade dos ambientes onde o projecto decorre e a dimensão e complexidade do mesmo.

Para o gestor de projectos é essencial a compreensão dos objectivos de cada projecto em relação a resultados expectáveis, âmbito, custos, tempo e níveis de qualidade e as interligações e dependências entre eles. Não menos importante é também o

entendimento da importância da vertente humana (liderança, equipas, conflitos, negociações…) no sucesso de um projecto.

No âmbito das tecnologias de informação e comunicação (TIC) a engenharia de software profissional está sujeita a restrições de orçamento e planeamento como as demais engenharias. As falhas ocorridas em vários sistemas estavam relacionadas com problemas de gestão da produção de software, nomeadamente a entrega atrasada, produtos não confiáveis, desempenho fraco e custos elevados.

A diferença entre a engenharia de software e outras engenharias destaca-se ao nível do produto, que se considera ser intangível, e ao nível do processo, dada a inexistência de um processo padrão. Normalmente os projectos de software de grande dimensão são frequentemente projectos que não se repetem, e com características únicas.

As principais actividades de gestão na engenharia de software estão relacionadas com a preparação de proposta do produto, a definição de objectivos e o modo como o projecto vai ser conduzido. O cálculo do orçamento do projecto, com a definição dos recursos necessários, bem como o planeamento e escalonamento do projecto, nomeadamente ao nível do estabelecimento de "*milestones*" (final de uma actividade) e "*deliverables*" (um resultado entregue ao cliente no fim de uma fase do projecto) são também actividades fundamentais na gestão de projectos informáticos.

Na gestão de recursos, a selecção e avaliação de pessoal terão em conta o orçamento estabelecido, a disponibilidade dos recursos e ainda o treino e competências específicas. Esta actividade é particularmente crítica, pois pode não ser possível seleccionar as pessoas ideais para trabalhar no projecto, porque o orçamento estabelecido pode não permitir a utilização de profissionais altamente especializados; concomitantemente os profissionais com a experiência apropriada podem não estar disponíveis. Os gestores têm que decidir dentro destas restrições de recursos, resultando que o planeamento de um projecto de software é uma actividade que consome considerável tempo de gestão.

A partição do projecto em tarefas e a estimação do tempo e recursos necessários para completar cada tarefa; a organização das tarefas num formato concorrente para fazer o melhor uso da força de trabalho; e a minimização das dependências entre as tarefas para evitar atrasos, são as funções do gestor de projectos que dependem fortemente da intuição e experiência. Deste modo, a gestão de projectos de software representa um desafio constante nas empresas.

O adequado controlo das várias variáveis de gestão tais como o âmbito, o custo, o prazo, a qualidade e o risco necessitam de uma base de técnicas e competências que não são fáceis de gerir. Uma perspectiva prática da gestão de projectos de software, pode ser adquirida nos modelos PMBOK (*Project Management Body of Knowledge - http://www.pmi.org/*) e PRINCE (*PRojects IN Controlled Environments - http://www.prince-officialsite.com/home/home.asp*).

A Figura 1 esquematiza um projecto composto por 12 tarefas, com as suas durações, dependências e respectivos *milestones*.

Tarefa	Duração (dias)	Dependencias
T1	8	
T2	15	
T3	15	T1 (M1)
T4	10	
T5	10	T2, T4 (M2)
T6	5	T1, T2 (M3)
T7	20	T1 (M1)
T8	25	T4 (M5)
T9	15	T3, T6 (M4)
T10	15	T5, T7 (M7)
T11	7	T9 (M6)
T12	10	T11 (M8)

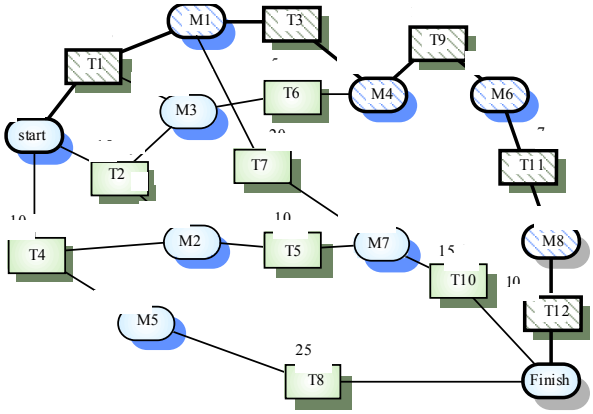

Figura 1: Projecto de Software

Este artigo apresenta uma abordagem para a alocação óptima de recursos, em redes de actividades, em condições de complementaridade de recursos, que ocorrem frequentemente na indústria do software, em projectos de engenharia de software.

O conceito de complementaridade, que tem sido discutido sob um ponto de vista económico (Kemer, 1993) pode ser incorporado no domínio da engenharia como um aumento da eficácia de um recurso principal (recurso-*P*) através da adição de um recurso de suporte (recurso-*S*). Aspectos relacionados com a melhoria do desempenho, redução da duração e aumento da qualidade, bem como o efeito do recurso de suporte no custo do projecto, têm sido apresentados por Silva, Tereso e Oliveira (2010).

A questão pode ser formulada como: Que quantidade de recursos principais (recursos-*P*) e de suporte (recursos-*S*) devem ser alocados às actividades do projecto, para atingirmos os melhores resultados, de forma mais económica?

2. Descrição do Problema

Considere uma rede de actividades nos vértices (nodos) (AoN) representada por $G=(N,A)$ com o conjunto de vértices $|N|=n$ (representando as "actividades") e o conjunto de arcos $|A|=m$ (representando a relação de precedência entre as actividades). Em geral, cada actividade requer o uso simultâneo de vários recursos (Tereso, Araújo, & Elmaghraby, 2008), (Tereso, Araujo, Moutinho, & Elmaghraby, 2009), (Tereso, Araújo, Moutinho, & Elmaghraby, 2009b).

Existe um conjunto de recursos "principais", denotados por *P*, com $|P|=\rho$. Tipicamente, um recurso principal tem várias unidades disponíveis (p.e. trabalhadores, máquinas, processadores, etc.) (Mulcahy, 2005). Adicionalmente, existe um conjunto de recursos de suporte, representados pela letra *S*, com $|S|=\sigma$ (tais como trabalhadores menos qualificados, ou computadores e dispositivos electrónicos, etc.)

que devem ser utilizados em conjunto com os recursos principais para aumentar o desempenho destes últimos. O número de recursos de suporte utilizados numa actividade varia em função dos recursos principais necessários para a execução dessa actividade. O impacto sobre o recurso-*P* é representado utilizando a variável $0 < v\left(r_p, s_q\right) \leq 1$ que indica a fracção pela qual o recurso-*S* $\left(s_q\right)$ aumenta a performance do recurso-*P* $\left(r_p\right)$. Tipicamente, $v\left(r_p, s_q\right) \in \left[0{,}1\ ;\ 0{,}5\right]$.

Assume-se que o impacto dos recursos-*S* é aditivo: se um subconjunto $\left\{s_q\right\}_{q=1}^{v}$ de recursos-*S* é usado como suporte ao recurso-*P* r_p na actividade a, e somente uma unidade de cada recurso-*S* é utilizada, então o desempenho do recurso-*P* é aumentado para,

$$x_a\left(r_p, \left\{s_q\right\}_{q=1}^{\sigma}\right) = x_a\left(r_p\right) + \sum_{q=1}^{\sigma} v\left(r_p, s_q\right) \tag{1}$$

Com $w_a\left(r_p\right)$ representando o conteúdo de trabalho do recurso-*P* r_p na actividade a, o recurso principal $r_p \in P$ completaria a actividade a no tempo $y_a\left(r_p\right)$ (eq. 2). Se este recurso principal é melhorado pela adição do recurso de suporte, então o seu tempo de processamento é reduzido para $y_a\left(r_p, s_q\right)$ (eq. 3).

$$y_a\left(r_p\right) = \frac{w_a\left(r_p\right)}{x_a\left(r_p\right)} \tag{2}$$

$$y_a\left(r_p, s_q\right) = \frac{w_a\left(r_p\right)}{x_a\left(r_p, s_q\right)} \tag{3}$$

Uma actividade normalmente requer uma utilização simultânea de mais de um recurso-*P* para a sua execução. O problema coloca-se então do seguinte modo.

A que nível deve ser utilizado cada recurso principal e que recursos de suporte lhe devem ser adicionados para optimizar um dado objectivo?

O tempo de processamento de uma actividade é dado pelo valor máximo das durações que podem resultar de uma afectação específica a cada recurso (veja a discussão prévia sobre o cálculo da duração considerando múltiplos recursos em (Tereso, Araújo, & Elmaghraby, 2008) (Tereso, Araujo, Moutinho, & Elmaghraby, 2009), (Tereso, Araújo, Moutinho, & Elmaghraby, 2009b)).

$$y\left(a\right) = \max_{all\ r_p}\left\{y_a\left(r_p\right)\right\} \tag{4}$$

Para melhor se entender esta representação, considere-se um simples projecto em modo de representação AoN (Figura 2). Este projecto é constituído por três actividades, 1, 2 e 3. Para cada uma iremos assumir que é necessário a utilização de

quatro recursos-*P*; nem todos os recursos são requeridos por todas as actividades (veja Tabela 1).

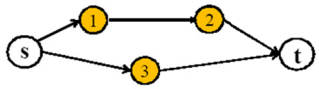

Figura 2: Projecto AoN com 3 actividades

Tabela 1: Conteúdo de trabalho (em homens-dia) das actividades do projecto 1

Recurso-*P* →	**1**	**2**	**3**	**4**
↓ **Actividade/Disponibilidade** →	**2**	**1**	**3**	**2**
A1	16	0	12	12
A2	0	7	10	8
A3	20	0	22	0

A relevância e o impacto dos recursos de suporte são representados na Tabela 2.

Tabela 2: A matriz P-S: Impacto dos recursos-*S* nos recursos-*P*

	Recurso-*P* →	**1**	**2**	**3**	**4**
↓ **Recurso-*S***	↓ **Disponibilidade**				
1	1	0,25	ϕ	0,25	ϕ
2	2	0,15	0,35	ϕ	ϕ

Neste problema considera-se o custo da utilização dos recursos, uma bonificação por se terminar mais cedo do que o previsto e uma penalização por se terminar mais tarde do que uma data prevista de término para o projecto. Foi já desenvolvido um modelo para minimizar o custo total do projecto, considerando que as actividades podem ser iniciadas logo que sejam sequenciáveis, se houver recursos principais suficientes para as iniciar (Silva, Tereso, & Oliveira, 2010). Alguns resultados foram também apresentados utilizando um procedimento baseado na análise da rede de actividades e no conceito estado (State Space) (Silva, Tereso, & Oliveira, 2010).

3. Algoritmo Genético

Uma vez que o Problema Job Shop (JSP) pode ser visto como um caso particular do problema da programação de projectos com restrições de recursos, do inglês "Resource Constrained Project Scheduling Problem" (RCPSP), iremos estender o Algoritmo Genético (AG) desenvolvido para o JSP (Oliveira, Dias, & Pereira, 2010) ao RCPSP, e particularmente para a programação de projectos em redes de actividades sob condições de complementaridade de recursos. O AG é baseado numa representação por

chaves aleatórias, a qual permite uma fácil reconfiguração para ser aplicada noutros problemas.

A simplicidade dos AG para modelar problemas complexos e a sua fácil integração com outros métodos de optimização foram os factores considerados para esta escolha. Inicialmente, o algoritmo proposto foi concebido para solucionar a versão clássica do JSP (Oliveira, 2007), mas é possível usar o mesmo método para solucionar outras variantes do JSP (Oliveira, 2006), ou neste caso, para solucionar também uma generalização do JSP que é o caso do RCPSP com complementaridade de recursos.

Uma das características que diferenciam os algoritmos genéticos convencionais é o facto de o algoritmo não tratar directamente com as soluções do problema, mas sim com uma representação da solução, o cromossoma. As manipulações do algoritmo são feitas sobre a representação e não directamente sobre a solução (Goldberg, 1989).

Representamos o problema da programação de projectos num gráfico AoN, porque é semelhante ao gráfico disjuntivo que é usado para representar o JSP (Roy & Sussmann, 1964). Uma actividade só pode ser iniciada se as precedentes estiverem concluídas e se o conjunto de todos os recursos principais necessários estiverem disponíveis. Um projecto tem restrições tecnológicas que determinam uma ordem específica para processar algumas das actividades e é necessário garantir que não haja sobreposição temporal, no processamento de tais actividades em termos de recursos comuns. Atendendo a esta característica, usamos o conceito de actividade sequenciável (uma actividade que pode ser iniciada) e em cada instante de decisão só é necessário escolher uma actividade do conjunto das actividades sequenciáveis. A escolha da actividade é dirigida pelo algoritmo genético respeitando os alelos existentes nos cromossomas que definem a prioridade de cada actividade.

Tradicionalmente, os algoritmos genéticos utilizavam cadeias binárias que formavam os cromossomas, que consistiam apenas em '0s' e '1s'. Alguns dos algoritmos genéticos mais recentes apresentam cromossomas mais específicos, para um dado problema, que resultam de um balanço do ganho de eficiência e da perda de alguma flexibilidade. Uma análise aos resultados mais recentes demonstra que o uso de cromossomas com valores reais frequentemente supera o uso de cromossomas binários. Outra alternativa de alfabeto ao código binário convencional é o código de Gray, que é também um sistema binário, onde dois valores sucessivos diferem em apenas um dígito (Goldberg, 1989).

O código de permutação era adequado para problemas de permutação. Neste tipo de representação, o cromossoma é um literal da sequência de operações nas máquinas. No caso JSP clássico, o cromossoma é composto por m subcromossomas, um para cada máquina, cada um composto por *n* genes, um para cada operação (Oliveira, 2007). O gene i do subcromossoma corresponde à operação processada na *i*-ésima posição na máquina correspondente. O alelo identifica o índice da operação no gráfico disjuntivo (Oliveira, 2007). Para as redes de actividades sob complementaridade de recursos, definimos um cromossoma com $n(\rho+\sigma+1)$ genes. Para cada actividade o cromossoma estabelece a quantidade de cada recurso-*P* e a quantidade de cada recurso-*S* complementar. Além disso, o cromossoma indica a prioridade de cada actividade.

Neste trabalho é utilizado o alfabeto de chaves aleatórias apresentado por Bean (1994). A característica mais importante das chaves aleatórias é que todos os descendentes gerados pelo cruzamento são soluções admissíveis, quando é usada em conjunto com um procedimento construtivo da solução baseado em todas as operações disponíveis para sequenciamento, e a prioridade é dada pelo alelo da chave aleatória. Através da dinâmica do algoritmo genético, o sistema aprende a relação entre vectores aleatórios e as soluções com bons valores da função objectivo. Outra vantagem da representação de chave aleatória é a possibilidade de se utilizar os operadores genéticos convencionais. Esta característica permite o uso do algoritmo genético noutros problemas de optimização, apenas adaptando algumas rotinas relacionadas com o problema.

Um cromossoma representa uma solução para o problema e é codificado como um vector de chaves aleatórias (números aleatórios). Neste trabalho, de acordo com Cheng, Gen, e Tsujimura (1996), a representação do problema pode ser entendida como uma mistura de uma representação baseada em regras de prioridade e na representação das chaves aleatórias.

O algoritmo genético tem uma estrutura muito simples e pode ser representado no algoritmo da Figura 3. Ela começa com a geração da população e a sua avaliação. Atendendo à aptidão dos cromossomas, os indivíduos são seleccionados para serem progenitores. O cruzamento é aplicado e é gerada uma nova população temporária, que também é avaliada. Comparando-se a aptidão dos novos elementos e dos seus progenitores, a população anterior é actualizada.

```
begin
P ← GerarPopulaçãoInicial()
Avaliar(P)
while condição de paragem não for alcançada do
    P' ← Recombinar(P)   // UX
    Avaliar(P')
    P ← Selecionar(P ∪ P')
end while
```

Figura 3: Algoritmo Genético

O Cruzamento Uniforme (UX) é usado neste trabalho. Este operador genético usa uma terceira sequência de números aleatórios e permuta os alelos de ambos progenitores se a chave aleatória é superior a um valor prefixado. A Figura 4 ilustra a aplicação de UX sobre dois progenitores (prnt1, prnt2) e troca os alelos, se a chave aleatória é maior ou igual a 0,75. Os genes 3, 4 e 16 são alterados o que dá origem a dois descendentes (dscndt1, dscndt2). O descendente 1 é semelhante ao progenitor prnt1, porque tem cerca de 75% dos seus genes.

	J1				J2				J3				J4			
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
prnt1	0,89	0,49	0,24	0,03	0,41	0,11	0,24	0,12	0,33	0,30	0,27	0,24	0,80	0,53	0,28	0,18
prnt2	0,83	0,41	0,40	0,04	0,29	0,35	0,38	0,01	0,42	0,32	0,28	0,28	0,72	0,61	0,44	0,13
randkey	0,64	0,72	0,75	0,83	0,26	0,56	0,28	0,31	0,09	0,11	0,37	0,25	0,23	0,05	0,40	0,76
dscndt1	0,89	0,49	0,40	0,04	0,41	0,11	0,24	0,12	0,33	0,30	0,27	0,24	0,80	0,53	0,28	0,13
dscndt2	0,83	0,41	0,24	0,03	0,29	0,35	0,38	0,01	0,42	0,32	0,28	0,28	0,72	0,61	0,44	0,18

Figura 4: O Cruzamento UX

4. Algoritmo Construtivo

As soluções são obtidas por um algoritmo, que é baseado no algoritmo de Giffler e Thompson (1960). Enquanto o algoritmo de Giffler e Thompson (1960) pode gerar todos os planos activos para o JSP, o algoritmo construtivo apenas gera o plano de acordo com o cromossoma. Como vantagens desta estratégia, salienta-se a menor dimensão do espaço de soluções, que no entanto inclui a solução óptima e o facto de o procedimento não produzir soluções impossíveis ou desinteressantes do ponto de vista da optimização. Por outro lado, uma vez que as dimensões do espaço de representação e do espaço de soluções são muito diferentes, esta opção pode representar um problema, porque muitos cromossomas podem representar a mesma solução.

O algoritmo construtivo tem n estágios e em cada estágio uma actividade é sequenciada. Para auxiliar na apresentação do algoritmo, considere a seguinte notação para o estágio t:

P_t - Plano parcial formado pelas $(t-1)$ actividades sequenciadas;

S_t - Conjunto de actividades sequenciáveis no estágio *t*, i.e. todas as actividades que devem preceder as actividades no conjunto S_t estão em P_t;

δ_k - A data mais cedo na qual a actividade a_k em S_t poderia ser iniciada. Este tempo representa a conclusão de todos os precedentes de a_k e a disponibilidade de todos os recursos que a_k irá usar (recursos principais e recursos de suporte);

φ_k - A data mais cedo que a actividade a_k em S_t pode ser finalizada, que é $\varphi_k = \delta_k + y_k$;

A^* - A actividade onde $\varphi^* = \min_{a_k \in S_k} \{\varphi_k\}$;

S_t^* - O conjunto de conflito formado por $a_k \in S_t$ que tem $\delta_k < \varphi^*$.

a_k^* - A actividade seleccionada para ser sequenciada no estágio t.

O algoritmo construtivo das soluções é apresentado na Figura 5.

Passo 1	Seja $t = 1$ com P_1 sendo nulo. S_1 será o conjunto formado por todas as actividades que não têm predecessoras, ou seja pelas actividades ligadas ao vértice início.
Passo 2	Identificar $\varphi^* = \min_{a_k \in S_k} \{\varphi_k\}$ e identificar A^*. Formar S_t^*.
Passo 3	Seleccionar a actividade a_k^* em S_t^* com o maior valor de alelo de prioridade.
Passo 4	Passar ao próximo estágio por: (1) adição de a_k^* a P_t, criando P_{t+1}; (2) remoção de a_k^* de S_t, criando S_{t+1} por adição (se existirem) a S_t das actividades directamente sucessoras de a_k^* e que tenham todos os seus predecessores em P_{t+1}; (3) aumentar *t* em 1.
Passo 5	Se existir alguma actividade ainda por sequenciar $(t < N)$, voltar ao *Passo 2*. Senão, Parar.

Figura 5: Algoritmo Construtivo

O formato utilizado é similar ao utilizado por Cheng, Gen e Tsujimura (1996), para apresentar o algoritmo de Giffler e Thompson (1960). No *Passo 3*, em vez de se usar uma prioridade estabelecida por uma regra de despacho, é usada a informação dada pelo cromossoma. Se o maior valor dos alelos for igual para duas ou mais actividades, então é escolhida aleatoriamente uma dessas actividades.

Considere o exemplo apresentado na Figura 2, Tabela 1 e Tabela 2, com três actividades (A_1, A_2, A_3). Nesta instância existem $\rho = 4$ recursos principais e $\sigma = 2$ recursos de suporte. Um cromossoma para representar uma solução para esta instância tem 21 genes, porque existem seis genes para cada actividade para estabelecer o número de elementos de cada recurso que serão usados, mais o gene A_k que define a prioridade da actividade. A Figura 5 apresenta o cromossoma de valores de chaves aleatórias desta instância. Os valores são gerados aleatoriamente entre 0 e 99.

A1	P1	P2	P3	P4	S1	S2	**A2**	P1	P2	P3	P4	S1	S2	**A3**	P1	P2	P3	P4	S1	S2
94	51	88	76	52	23	68	**36**	73	60	61	53	75	35	**47**	7	15	42	86	16	16

Figura 6: Um cromossoma

A actividade A_1 tem uma prioridade de 94, a maior, enquanto A_2 tem a menor prioridade (36). Para definir o número de elementos de cada recurso, utilizamos a Tabela 1. A disponibilidade do recurso-*P*3 é de 3 unidades. Definimos três intervalos iguais entre 0 e 99. Para este recurso, se o alelo é um valor entre 0 e 32 é alocada 1 unidade, para valores entre 33 e 66, são alocadas 2 unidades e para valores de alelos entre 67 e 99, serão alocadas 3 unidades. Para estabelecer o número de unidades para os recursos de suporte, o procedimento é similar, mas também é incluída a possibilidade de se alocar 0 unidades de recursos de suporte, porque não é obrigatório

o uso destes recursos. Respeitando esta regra, o cromossoma apresentado na Figura 6 define as seguintes quantidades de recursos a serem utilizadas, que são apresentadas na Figura 7.

A1	P1	P2	P3	P4	S1	S2	A2	P1	P2	P3	P4	S1	S2	A3	P1	P2	P3	P4	S1	S2
Units	2	0	3	2	0	2	Units	0	1	2	2	1	1	Units	1	0	2	0	0	0

Figura 7: Quantidade de unidades de recursos a ser utilizada

São alocadas zero unidades de um recurso-*P* a uma actividade, se a actividade não utiliza esse recurso principal, que é o caso de P_2 na actividade A_1, de acordo com a Tabela 1.

A alocação dos recursos de suporte para os recursos principais é feita respeitando a quantidade de trabalho existente depois da alocação dos recursos principais. A primeira unidade é alocada ao recurso-*P* com maior quantidade de trabalho. Após a alocação, a quantidade de trabalho é recalculada e então é feita a próxima alocação e assim sucessivamente. A actividade A_1 apresenta o conteúdo de trabalho, indicado na Tabela 3.

Tabela 3: Conteúdo de trabalho da Actividade A1 (em homens-dia)

***P*-resource →**	**1**	**2**	**3**	**4**
A1	16	0	12	12

Alocando as unidades de recursos principais definidas pelo cromossoma a duração da actividade A_1 é a indicada na Tabela 4.

Tabela 4: Duração da Actividade A1 após afectação dos recursos principais

***P*-resource →**	**1**	**2**	**3**	**4**
A1	8	0	4	6

A primeira unidade de recurso de suporte S_2 é alocada a P_1. Recalculando a duração da actividade A_1 usando (3), tem-se os valores apresentados na Tabela 5.

Tabela 5: Duração da Actividade A1 após afectação de uma unidade de recurso de suporte

***P*-resource →**	**1**	**2**	**3**	**4**
A1	7,442	0	4	6

A segunda unidade de recursos de suporte S_2 é alocada a P_1. Recalculando a duração da actividade A_1 usando (3), tem-se os valores apresentados na Tabela 6.

Tabela 6: Duração da Actividade A1 após afectação de duas unidades de recurso de suporte

***P*-resource →**	**1**	**2**	**3**	**4**
A1	6,957	0	4	6

Aplicando o mesmo procedimento, as durações de todas as actividades são apresentadas na Tabela 7.

Tabela 7: Duração das Actividades

***P*-resource →**	**1**	**2**	**3**	**4**
A1	6,957	0	4	6
A2	0	5,185	4,444	4
A3	20	0	11	0

5. Exemplo Numérico

A Figura 8 ilustra os resultados da aplicação do algoritmo construtivo. Apresenta-se a evolução da programação do conjunto S_t e o correspondente tempo de início e o tempo de conclusão para cada actividade durante a execução do algoritmo construtivo. O gráfico de Gantt final do projecto também é apresentado e mostra a ocupação de todas as unidades de recursos.

S_1	1	3
δ_k	0	0
φ_k	6,957	20
a_k^*	1	

S_2	2	3
δ_k	6,957	6,957
φ_k	12,14	26,96
a_k^*		3

S_3	2
δ_k	17,97
φ_k	23,14
a_k^*	2

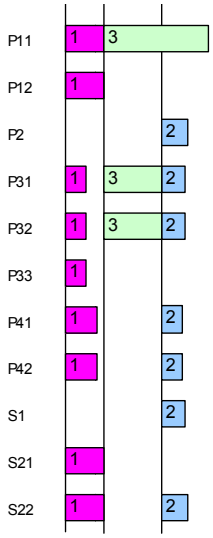

Figura 8: Execução do algoritmo construtivo

Considere a data final do projecto igual a 24 unidades de tempo, e os seguintes parâmetros:

- Cada unidade de recurso principal custa 4 unidades monetárias (UM) por unidade de conteúdo de trabalho;
- Cada unidade de recurso de suporte custa 1 UM por unidade de conteúdo de trabalho;
- O custo do atraso é de 60 UM por unidade de tempo;
- O custo (bónus) pelo término mais cedo é -40 UM por unidade de tempo.

O projecto termina no instante de tempo 26,96, com 2,96 unidade de atraso. O custo dos recursos é igual a 845 UM e o custo do atraso é de 177,39 UM. O custo total da solução é 1022,39 UM.

6. Conclusão

O objectivo deste artigo foi fornecer um modelo formal para alguns problemas não resolvidos na gestão de projectos, especialmente no que diz respeito à utilização de recursos de suporte, e apresentar a sua implementação num algoritmo genético. A importância do problema reside na oportunidade de se desenvolver um sistema que permita não só melhorar a afectação de recursos frequentemente escassos, mas também resultar numa redução das incertezas dentro dos projectos, combinada com o aumento da performance e com a redução do custo do projecto. O modelo foi inicialmente apresentado em Silva, Tereso e Oliveira (2010) mas faltava a sua implementação e aplicação a algumas redes de actividades, para demonstrar a sua validade. Depois foi apresentado o procedimento desenvolvido para solucionar o problema descrito pelo modelo matemático tendo sido aplicado a duas redes de actividades simples, obtendo assim os resultados desejados, através de uma implementação inicial em C (Silva, Tereso, & Oliveira, 2010).

Considerando a viabilidade do modelo proposto, acreditamos que ele pode fornecer ao utilizador uma nova opção de planeamento para determinar a melhor combinação de recursos e o menor custo do projecto, melhorando a capacidade de planeamento das empresas.

Este artigo apresenta uma estrutura para implementar um algoritmo genético para solucionar o problema do planeamento de projectos em redes de actividades em condições de complementaridade de recursos. O plano é construído usando informações fornecidas através do algoritmo genético para ordenar as actividades. Apresentamos um exemplo da aplicação do modelo e obtivemos os resultados preliminares para um pequeno projecto.

Na sequência deste estudo pretendemos testar o procedimento apresentado num conjunto de instâncias disponíveis na literatura de RCPSP às quais serão adicionados os recursos de suporte.

8. Referências

Bean, J. (1994). Genetics and random keys for sequencing and optimization. *ORSA Journal on Computing* , pp. 154–160.

Cheng, R., Gen, M., & Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms - I. *Representation, Computers & Industrial Engineering*, (pp. 983-997).

Giffler, B., & Thompson, G. L. (1960). Algorithms for solving production scheduling problems. *Operations Research, 8* , pp. 487-503.

Goldberg, D. E. (1989). Genetic Algorithms in Search. *Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Kemer, M. (1993). The O-Ring Theory of Economic Development. *The Quarterly Journal of Economics, 108*, 551-575.

Mulcahy, R. (2005). *PMP, Exam Prep – Rita's Course in a Book for Passing the PMP Exam Fifth Edition – For PMBOK Guide – Third Edition*. (5a Ed. ed.). USA: RMC Publications.

Oliveira, J. (2006). A genetic algorithm with a quasi-local search for the job shop problem with recirculation. *Applied Soft Computing Technologies: The Challenge of Complexity*, (pp. 221-234). Springer, Berlin / Heidelberg.

Oliveira, J. (2007). Scheduling the truckload operations in automatic warehouses. *European Journal of Operational Research*.

Oliveira, J., Dias, L., & Pereira, G. (2010). Solving the Job Shop Problem with a random keys genetic algorithm with instance parameters. *2nd International Conference on Engineering Optimization*. Lisbon – Portugal.

Roy, B., & Sussmann, B. (1964). Les problèmes d'ordonnancement avec contraintes disjonctives. Paris.

Silva, H., Tereso, A., & Oliveira, J. (2010). On Resource Complementarity in Activity Networks - Further Results. *2nd International Conference on Engineering Optimization*. Lisbon - Portugal.

Silva, H., Tereso, A., & Oliveira, J. (2010). On Resource Complementarity. *3rd International Conference on Information System, Logistics and Supply Chain*. Casablanca - Morocco.

Tereso, A. P., Araujo, M., Moutinho, R., & Elmaghraby, S. (2009). Duration Oriented Resource Allocation Strategy on Multiple Resources Projects under Stochastic Conditions. *International Conference on Industrial Engineering and Systems Management*. Montreal - Canada.

Tereso, A., Araújo, M., & Elmaghraby, S. (2008). Project management: multiple resources allocation. *International Conference on Engineering Optimization*. Rio de Janeiro - Brazil.

Tereso, A., Araújo, M., Moutinho, R., & Elmaghraby, S. (2009b). Quantity Oriented Resource Allocation Strategy on Multiple Resources Projects under Stochastic

Conditions. *International Conference on Industrial Engineering and Systems Management*. Montreal - Canada.