

RESOLUÇÃO DO PROBLEMA DE PROGRAMAÇÃO DE PROJETOS COM RECURSOS RESTRITOS E MULTI-HÁBEIS UTILIZANDO A META-HEURÍSTICA BRKGA

Flavio Augusto Barbosa Rodrigues

Instituto Tecnológico de Aeronáutica

Praça Marechal Eduardo Gomes, nº 50, Vila das Acácias, São José dos Campos-SP

flavioaugustofabr@fab.mil.br

Antonio Augusto Chaves

Universidade Federal de São Paulo, Unidade Parque Tecnológico

Avenida Cesare Mansueto Giulio Lattes, nº 1.201, Eugênio de Mello, São José dos Campos-SP

antonio.chaves@unifesp.br

RESUMO

O presente artigo apresenta um algoritmo baseado na meta-heurística BRKGA para resolução do problema de programação de projetos com recursos restritos e multi-hábeis (MS-RCPSP), com aplicação futura no planejamento de tarefas em escritórios de engenharia do setor de construção civil. São empregadas instâncias do banco iMOPSE para realização dos testes e as soluções obtidas são comparadas com cinco algoritmos divulgados na literatura. O algoritmo alcança os resultados do estado da arte em 61% das instâncias testadas, bem como iguala ou melhora a solução em todas as instâncias quando comparado com três algoritmos de referência. Os resultados indicam a aplicabilidade futura do algoritmo desenvolvido em instâncias reais do problema sob estudo, etapa em desenvolvimento na presente pesquisa.

PALAVRAS-CHAVE: BRKGA. Projetos de engenharia. Construção civil.

Tópicos: meta-heurística.

ABSTRACT

This article presents an algorithm based on the BRKGA meta-heuristic for solving the multi-skill resource constrained project scheduling problem (MS-RCPSP), to future planning activities of engineering offices in the civil construction sector. Instances of the iMOPSE database are used to carry out the tests and the obtained solutions are compared with five algorithms published in the literature. The algorithm achieves state-of-the-art results in 61% of the tested instances, as well as equals or improves the solution in all instances when compared to three reference algorithms. The results indicate the future applicability of the developed algorithm in real instances of the problem under study, research stage at development.

KEYWORDS. BRKGA. Engineering projects. Construction sector.

Paper topics: metaheuristic.

1. Introdução

Arquitetura, Engenharia e Construção (AEC) refere-se à tarefa laboral dedicada aos projetos e obras de construção civil. Inclui os estudos e projetos, construção ou reforma, de empreendimentos dos mais diversos tipos e complexidade: uma casa ou um arranha-céu de escritórios; pontes, estradas, portos e aeroportos; ou a infraestrutura de gás, água, energia e telecomunicações, essencial para a tarefa humana atualmente. Contudo, para que um empreendimento qualquer saia do campo das ideias e se concretize, faz-se necessário envia esforços para uma importante etapa do ciclo de vida desses bens: a fase de projeto.

Segundo definição de mercado, um projeto consiste em esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo, no caso em questão, a entrega do objeto que se deseja, com todas as suas funcionalidades. O projeto de AEC, de construção civil ou de engenharia, é constituído de diversas etapas, as quais encadeiam-se entre si, gerando entregáveis: na forma de desenhos técnicos, como plantas, vistas e detalhes; cadernos de especificações técnicas de materiais e técnicas construtivas a serem aplicadas; tabelas e memoriais de cálculo que demonstram como se chegou àquelas definições de projeto; bem como listas detalhadas com descrição qualitativa e quantitativa dos materiais e serviços que comporão a obra, acompanhada de seu orçamento de mercado. Cada um dos referidos entregáveis deve ser produzido por profissional competente e habilitado legalmente para a tarefa, como arquitetos e engenheiros civis, eletricitas, mecânicos, bem como desenhistas, topógrafos e orçamentistas.

Deve-se, ainda, destacar que cada uma dessas etapas de projeto possui uma relação seriada ou paralela, de modo que determinada etapa só pode se iniciar quando da conclusão de uma etapa predecessora, de maneira que há um fluxo de informações, desenhos e especificações por entre as disciplinas de projeto, formando uma rede ou grafo de precedência. Exemplificando, as etapas descritas, a Figura 1 ilustra a representação de rede de precedência de um projeto hipotético de engenharia.

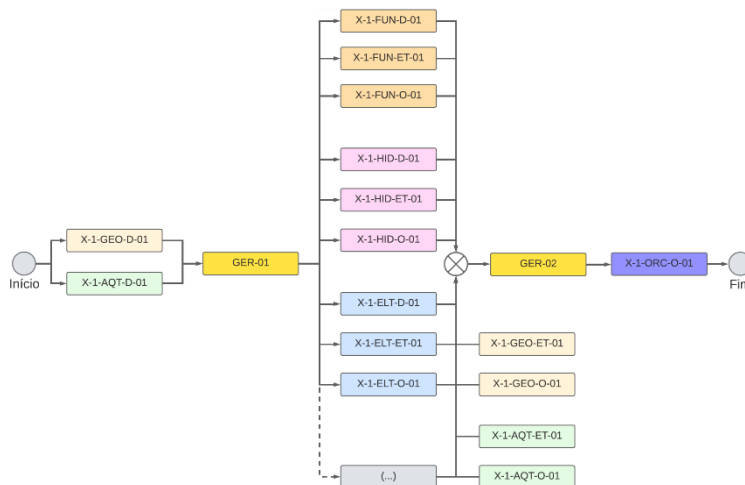


Figura 1 – Grafo de precedência, extraído da Estrutura Analítica de um projeto hipotético de construção civil, adaptado de [Hamilton 2008].

Em escritórios de AEC, como de empresas especializadas em projetos ou em construtoras, usualmente compartilha-se uma equipe de profissionais com uma carteira de projetos, de modo que cada tarefa de cada projeto passa a competir pelos profissionais, pela sua designação visando execução do projeto, dado determinado horizonte de planejamento. Usualmente, há uma finita equipe de recursos humanos disponíveis para condução de projetos concorrentes, de modo que ferramentas como o método do caminho crítico ou PERT/CPM não contemplam tal escassez de recursos [Gonçalves et al. 2008]. Dessa forma, faz-se necessário recorrer a modelagens que venham a considerar tais peculiaridades do problema, de modo a se oferecer aos setores de planejamento e programação de projetos de tais escritórios soluções automatizadas e de boa

qualidade, para além do que um gerente de projetos experiente poderia obter utilizando-se manualmente de aplicativos comerciais.

O problema em questão refere-se à programação de projetos, *project scheduling problems* (PSP), que se trata da classe de problemas que objetiva determinar a alocação de recursos a tarefas em dado horizonte de planejamento, com vistas à otimização de alguma métrica, relacionada ao tempo de execução das tarefas, seu custo associado ou até de múltiplos objetivos [Baker e Trietsch 2018]. Nesse contexto, em cenários de limitação de recursos, em que metodologias como o PERT/CPM se mostram pouco eficazes, surge a definição do problema de programação de projetos com recursos restritos, ou *resource constrained project scheduling problem* (RCPSP). No modelo generalista do RCPSP, as tarefas a serem programadas possuem restrições de precedência, acompanhadas da capacidade de cada tipo de recurso, visando à minimização do tempo de conclusão das tarefas ou *makespan*. Contudo, de modo a refletir as peculiaridades da problemática em situações reais, existem diversas variantes para o problema em questão, a depender das características dos recursos, das tarefas, da função objetivo e dos dados disponíveis para modelagem [Hartmann e Briskorn 2021].

Da análise das características do problema real sob estudo, verifica-se a aplicabilidade da modelagem do problema de programação de projetos com recursos restritos multi-habilidade, ou *multi-skill resource constrained project scheduling problem* (MS-RCPSP), extensão do problema clássico de literatura, caracterizada por camada adicional de complexidade, afeta às habilidades que cada tarefa requer e que os recursos podem ou não possuir. Nessa variante, uma dada tarefa requer que um determinado recurso possua uma dada habilidade para sua execução, de modo que as tarefas passam a concorrer por habilidades dominadas por recursos e não mais pelos recursos propriamente ditos. A Figura 2 contém uma ilustração de um problema-instância, o qual pretende-se empregar no planejamento de tarefas de desenvolvimento de projetos de construção civil, em etapa futura da presente pesquisa. Os recursos humanos possuem habilidades e cada tarefa requer um profissional com determinada habilidade.

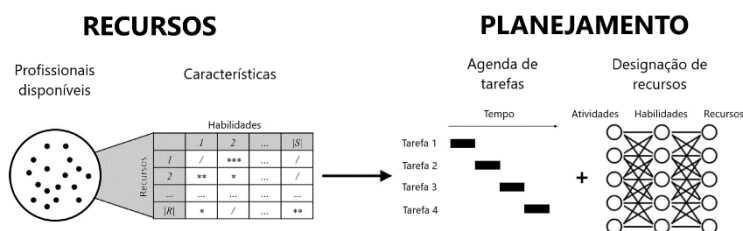


Figura 2 – Problema-instância do MS-RCPSP para programação de projetos de engenharia, adaptado de [Snauwaert e Vanhoucke 2021].

A Tabela 1 descreve uma lista não extensiva de publicações de literatura em que se estudou o emprego do MS-RCPSP, acompanhado do método de resolução adotado e do banco de instâncias empregado. Foram destacadas as publicações cuja função objetivo adotada tenha sido a minimização do tempo de conclusão do projeto. Observa-se vasta gama de métodos registrados: heurísticas construtivas baseadas em regras de prioridade; obtenção de limitantes inferiores para o problema; emprego de métodos exatos, como a resolução de modelos matemáticos via solucionadores comerciais, bem como a aplicação dos métodos *Branch&Bound* e *Branch&Price*; emprego de meta-heurísticas, predominantemente métodos populacionais, com destaque para os algoritmos genéticos e variantes. Cabe salientar que o MS-RCPSP apresenta-se como um modelo matemático de aplicação na área de projetos, não tão somente no setor da construção civil, bem como no desenvolvimento de aplicativos de tecnologia da informação, nos segmentos de geração, transmissão e distribuição de energia, telecomunicações, setor de serviços e consultorias diversas [Snauwaert e Vanhoucke 2021]. Para consulta do leitor, o trabalho de [Snauwaert e Vanhoucke

2023] apresenta uma ampla revisão de trabalhos relacionados ao MS-RCPSP, suas variantes e aplicações diversas.

Dessa maneira, o presente artigo tem por objetivo apresentar uma proposta de algoritmo para resolução do MS-RCPSP clássico, com instâncias de literatura, visando aferir seu desempenho e realizar sua validação. Posteriormente, em trabalhos futuros, objetiva-se gerar uma instância baseada em dados reais e específicos para projetos de engenharia e arquitetura, com vistas à sua resolução e comparação de resultados frente ao planejamento de tarefas manual atualmente realizado. Para tal, considerando a característica do problema de otimização combinatória, propõe-se para resolução do problema o desenvolvimento de uma meta-heurística baseada no BRKGA, bem como sua aplicação em instâncias da literatura que sejam representativas do problema sob estudo. Desse modo, o presente artigo pretende colaborar com a lacuna de pesquisa identificada, de escassez de trabalhos associados ao planejamento de tarefas em escritórios de engenharia e arquitetura, fazendo uso de ferramentas de otimização combinatória, como o algoritmo baseado na meta-heurística BRKGA, proposto no presente trabalho.

Autores	Método/Algoritmo	Instâncias	N	K	L	Eficiência
[Correia et al. 2012]	PLIM e Heurística	Própria (PSPLIB)	20	10 a 30	4	Não considera
[Montoya et al. 2014]	Branch & Price	Própria (PSPLIB)	20	10 a 30	4	Não considera
[Myszkowski et al. 2015]	ACO híbrido	Própria (iMOPSE)	100 a 200	5 a 40	9 a 15	Considera
[Almeida et al. 2016]	Heurística	[Correia et al. 2012] e própria	20 a 40	10 a 60	4	Não considera
[Myszkowski e Siemieniowski 2016]	GRASP	iMOPSE	100 a 200	5 a 40	9 a 15	Considera
[Young et al. 2017]	CP	Própria	22 a 62	5 a 60	2 a 15	Não considera
[Almeida et al. 2018]	GA	[Correia et al. 2012] e própria	20 a 40	10 a 60	4	Não considera
[Myszkowski et al. 2018]	DEGR híbrido	iMOPSE	100 a 200	5 a 40	9 a 15	Considera
[Laszczyk e Myszkowski 2019]	GA (NSGA-II)	iMOPSE	100 a 200	5 a 40	9 a 15	Considera
[Lin et al. 2020]	GP-HH	iMOPSE	100 a 200	5 a 40	9 a 15	Considera
[Snauwaert e Vanhoucke 2021]	GA híbrido	Própria	30	8 a 216	4	Considera
[Jedrzejowicz e Ratajczak-Ropel 2022]	ATMAS	iMOPSE	100 a 200	5 a 40	9 a 15	Considera
[Quoc et al. 2022]	PSO modificado	iMOPSE	100 a 200	5 a 40	9 a 15	Considera

Tabela 1 – Trabalhos publicados com resolução do MS-RCPSP, com objetivo de minimização da duração do projeto, adaptado de [Snauwaert e Vanhoucke 2021].

O presente Artigo é organizado da seguinte maneira: na Seção 2 descreve-se o problema sob estudo, com o auxílio da modelagem matemática do problema; a Seção 3 expõe a metodologia científica aplicada, como o algoritmo elaborado para resolução do problema; a Seção 4 apresenta as instâncias da literatura e equipamento utilizados nos testes computacionais, bem como os resultados obtidos; e, por fim, a Seção 5, exprime as considerações finais, bem como propostas para extensão da pesquisa e trabalhos futuros.

2. Descrição do problema

O modelo matemático de programação linear inteira mista extraído de [Correia et al. 2012] representa-se por um grafo orientado $G = (V, E)$ no qual as tarefas de $V = \{0, \dots, j, n + 1\}$ representam os nós e, portanto, cada uma das n tarefas. As tarefas $j = 0$ e $j = n + 1$ são fictícias, sendo empregadas apenas para representar, respectivamente, um marco de início e de fim do projeto. Por sua vez, os arcos $(i, j) \in E$ representam as relações de precedência, de modo que indicam que uma dada tarefa i precede a tarefa j . A cada tarefa V associa-se um custo, definido como p_j , relativo à duração da tarefa em unidades de tempo. Por sua vez, define-se ainda o conjunto A como sendo o conjunto dos pares de tarefas (i, j) que não possuem relação de precedência entre si e, portanto, não possuem arcos contidos em E . Há, ainda, os conjuntos de recursos $R = \{0, \dots, k, \dots, K\}$ e de habilidades $L = \{0, \dots, l, \dots, L\}$, que definem as quantidades dos K recursos existentes, bem como o número L de habilidades existentes no problema.

Dos conjuntos acima, procedem, ainda, os seguintes conjuntos do problema-instância: L_j , conjunto de habilidades L requeridas por cada uma das j tarefas; L_k , conjunto de habilidades dominadas por cada um dos K recursos disponíveis; V_k , conjunto de tarefas que requerem habilidades dominadas pelos K recursos; R_j , conjunto de recursos que podem contribuir ao menos com uma habilidade requerida por cada uma das j tarefas; R_l , conjunto de recursos que dominam uma dada habilidade l .

Cada tarefa j requer uma ou mais habilidades l para que seja executada. Para cada uma dessas habilidades, a quantidade de recursos necessários para realização da tarefa j é dada pelo parâmetro r_{jl} . Em casos especiais, em que uma dada tarefa j requer recurso único que domine uma habilidade l , caso habitual em problemas de projetos de engenharia, $r_{jl} = 1$. Analogamente, cada recurso k domina uma ou mais habilidades l . Contudo, cada recurso pode colaborar na execução de somente uma tarefa j em um mesmo instante de tempo. Interrupções de tarefas não são aceitáveis, de modo que uma vez iniciada, a tarefa deve ser concluída. Do mesmo modo, os recursos são considerados disponíveis durante toda a execução da tarefa, sendo possível previamente o registro das eventuais indisponibilidades dos recursos como dado de entrada.

Feitas essas considerações, as Equações (1) a (10) referem-se ao modelo do MS-RCPSP de [Correia et al. 2012]. O modelo em questão possui três variáveis de decisão: Q_j , que define o instante de tempo em que cada tarefa j se inicia; u_{ij} , cujo valor denota, para os pares $(i, j) \in A$ se a tarefa i termina antes do início da tarefa j ($u_{ij} = 1$), ou $u_{ij} = 0$ caso contrário; w_{jlk} , que relaciona a associação do recurso k à tarefa j empregando a habilidade l , caso em questão em que $w_{jlk} = 1$, assumindo valor igual a zero caso contrário.

$$\min Q_{n+1} \quad (1)$$

sujeito a:

$$Q_j \geq Q_i + p_i \quad \forall (i, j) \in V \cap (i, j) \in E \quad (2)$$

$$Q_j \geq Q_i + p_i - M(1 - u_{ij}) \quad \forall (i, j) \in V \cap (i, j) \in A \quad (3)$$

$$u_{ij} + u_{ji} \leq 1 \quad \forall (i, j) \in V \cap (i, j) \in A \mid i < j \quad (4)$$

$$\sum_{k \in R_l} w_{jlk} = r_{jl} \quad \forall j \in V, l \in L_j \quad (5)$$

$$\sum_{l \in L_k \cap L_j} w_{jlk} \leq 1 \quad \forall k \in R, j \in V_k \quad (6)$$

$$\sum_{l \in L_k \cap L_j} w_{ilk} + \sum_{l \in L_k \cap L_j} w_{jlk} \leq u_{ij} + u_{ji} + 1 \quad \forall k \in R, (i, j) \in V_k \cap (i, j) \in A \quad (7)$$

$$ES_j \leq Q_j \leq LS_j \quad \forall j \in V \quad (8)$$

$$w_{jlk} \in \{0, 1\} \quad \forall k \in R, j \in V_k, l \in L_k \cap L_j \quad (9)$$

$$u_{ij} \in \{0, 1\} \quad \forall (i, j) \in V \cap (i, j) \in A \quad (10)$$

A função objetivo da Equação (1) estabelece o propósito de minimização do *makespan* ou duração total do projeto. Por meio de (2), determina-se o atendimento da relação de precedência entre os pares de tarefas que se relacionam. As Equações (3) e (4) definem a relação entre os instantes de início de cada par de tarefas sem relação de precedência estabelecida. Em (5), institui-se o atendimento integral das demandas por recursos de cada tarefa, nas habilidades necessárias. Pela Equação (6), restringe-se o emprego de cada recurso ao atendimento de somente uma habilidade por tarefa. Em conjunto com (7), define-se ainda que cada recurso pode atuar somente

em uma tarefa em dada janela de tempo. As restrições (8) prescrevem que as tarefas possuem um intervalo para sua execução, considerando que as tarefas são encadeadas. Cada tarefa não pode ser executada em qualquer intervalo de tempo, dado o requisito que as tarefas predecessoras tenham sido realizadas antes. Assim, define-se ES_j e LS_j como, respectivamente, o instante de tempo mais cedo e mais tarde que uma dada tarefa pode iniciar. Por fim, (9) e (10) asseguram o domínio das variáveis de decisão empregadas no modelo matemático.

3. BRKGA

Para resolução do MS-RCPSP, propõe-se o emprego de uma meta-heurística, desenvolvida com base no algoritmo genético de chaves aleatórias enviesadas ou *biased random-key genetic algorithm* (BRKGA), proposto por [Gonçalves e Resende 2011]. Trata-se de uma variação do algoritmo genético clássico, cujo processo evolutivo do BRKGA original encontra-se ilustrado pela Figura 3.

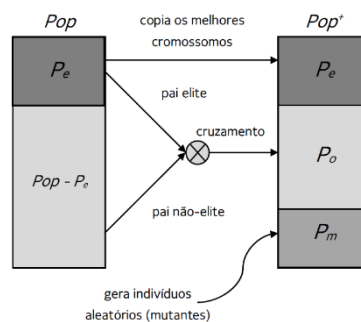


Figura 3 – Diagrama do BRKGA clássico.

Nessa meta-heurística populacional, o processo evolutivo consiste na geração de uma população inicial, Pop , cuja quantidade de cromossomos ou indivíduos depende de parâmetro p , que define o tamanho da população. Cada indivíduo possui n genes, cujo valor de n depende da representação de solução adotada para o problema e definida para o método de resolução. Em cada gene de cada cromossomo, emprega-se uma chave aleatória, ou seja, valores variando no intervalo $[0, 1]$. Os indivíduos da população são decodificados por um importante elemento de código chamado de decodificador ou *decoder*. Cabe a essa sub-rotina estabelecer a um dado cromossomo de chaves aleatórias uma solução factível para o problema sob estudo, a qual pode ser quantificada conforme a qualidade da solução ou valor de sua função objetivo, também chamada de *fitness*. Após decodificação dessa população inicial, os indivíduos são ordenados segundo seu *fitness*, do melhor para o pior. Selecionam-se os melhores elementos da população e constitui-se um subconjunto da população, chamado de população elite, Pop_e , em tamanho definido pelo parâmetro p_e , tamanho da população elite, uma porcentagem da população integral. Aos demais indivíduos, não selecionados para a população elite, denomina-se população não-elite, a qual será empregada posteriormente para constituição da nova população.

De maneira iterativa, gera-se uma nova população, Pop^+ , a qual será composta por três subconjuntos de indivíduos: o primeiro subconjunto é uma cópia estrita dos indivíduos da população elite da geração anterior, Pop_e ; o segundo subconjunto denomina-se de elementos mutantes, P_m , que se referem a indivíduos gerados aleatoriamente em processo idêntico ao que se gerou a população inicial, exceto pela quantidade, definida pelo parâmetro de tamanho da população mutante, p_m ; por fim, a população é completada com indivíduos oriundos do cruzamento entre indivíduos da elite e da partição não elite da população anterior. A geração desses filhos se dá em processo conhecido como cruzamento parametrizado uniforme. Nesse procedimento, para formação de cada novo filho, sorteia-se para serem os genitores um indivíduo da população elite e um indivíduo da população não-elite da geração anterior. Para cada um dos genes do novo filho, sorteia-se um número aleatório e verifica-se se o valor dessa chave é inferior à probabilidade de herança de gene da elite, $p_e > 0,5$. Em caso positivo, o gene será herdado do

indivíduo oriundo da elite. Caso contrário, herda-se o gene do indivíduo da partição não elite. A Figura 4 ilustra a referida operação de cruzamento parametrizado uniforme de [Spears e Dejong 1991].

Cromossomo pai elite	0,0238	0,7375	0,9769	0,7995	0,2784
Cromossomo pai não-elite	0,8732	0,9159	0,8411	0,8712	0,3581
No. Aleatório: def. origem do gene filho	0,5828	0,3804	0,7009	0,8354	0,7531
Relação com $p_e = 0,7$	<	<	>	>	>
Cromossomo filho	0,8732	0,9159	0,9769	0,7995	0,2784

Figura 4 – Exemplo de geração de cromossomo pelo cruzamento uniforme parametrizado de [Spears e Dejong 1991].

Após geração da nova população, ordena-se novamente por meio do *decoder* tais indivíduos pela qualidade da solução. O processo de geração de uma nova população se repete e, a cada repetição, verifica-se se houve uma melhor solução encontrada, armazenando-a. Importante, ainda, destacar o processo de decodificação das chaves aleatórias em soluções factíveis para o problema sob análise, o qual encontra-se ilustrado pelas Figuras 5 e 6.

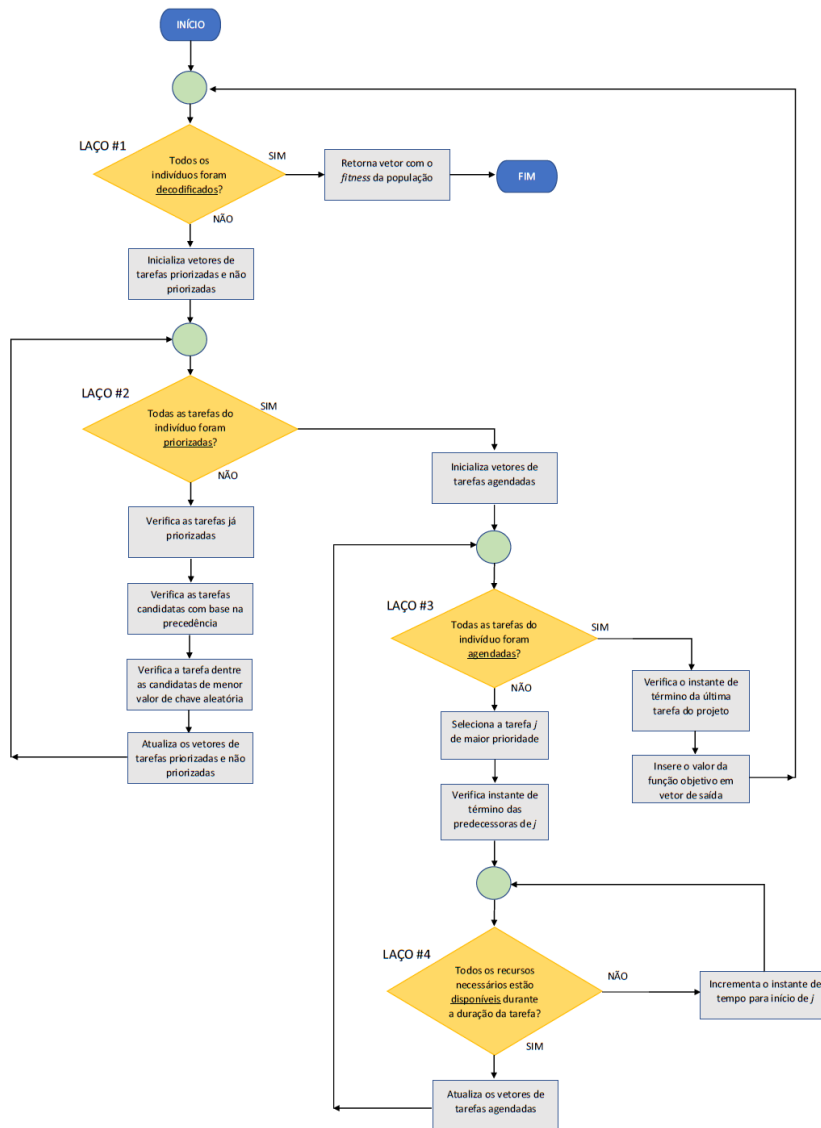


Figura 5 – Fluxograma do decodificador empregado no algoritmo.

Na Figura 5, consta fluxograma explicativo do funcionamento do *decoder*, responsável pela decodificação da população de cromossomos em soluções factíveis para o problema sob análise. O *decoder* itera até que cada elemento da população tenha sido devidamente decodificado (laço nº 1). Para cada indivíduo, com base no valor das chaves aleatórias, define-se qual será a ordem de prioridade de cada tarefa, respeitada também a precedência obrigatória entre as tarefas (laço nº 2), conforme processo exemplificado na Figura 6. De posse dessa ordem de prioridade, cada tarefa é agendada seguindo a prioridade definida e no primeiro instante de tempo possível, desde que haja recurso disponível durante toda a execução da tarefa (laços nº 3 e 4). Desse modo, realiza-se o agendamento das tarefas conforme a disponibilidade dos recursos, resultando na formação do planejamento e obtenção do valor da função objetivo, a saber, a duração do projeto.

Tarefa	Precedência	Lista de tarefas posteriores
1	0	4,5,6,7,10,11
2	0	6,7,10,11
3	0	8,9,12
4	1	5,10
5	4	10
6	1,2	7,10,11
7	6	10,11
8	3	9,12
9	8	12
10	5,7	-
11	7	-
12	9	-

- 1) Cromossomo de chaves aleatórias inicial, gerado para o exemplo.

1	2	3	4	5	6	7	8	9	10	11	12
0.45	0.30	0.70	0.55	0.35	0.15	0.90	0.65	0.10	0.05	0.20	0.25
- 2) Tarefas candidatas: 1, 2, 3
Verifica dentre tarefas candidatas a de menor chave aleatória: tarefa 2
Seleciona a tarefa prioritária e remove ela da lista

1	2	3	4	5	6	7	8	9	10	11	12
0.45	0.30	0.70	0.55	0.35	0.15	0.90	0.65	0.10	0.05	0.20	0.25

Ordem do schedule
[2]
- 3) Tarefas candidatas: 1, 3
Verifica dentre tarefas candidatas a de menor chave aleatória: tarefa 1
Seleciona a tarefa prioritária e remove ela da lista

1	3	4	5	6	7	8	9	10	11	12
0.45	0.70	0.55	0.35	0.15	0.90	0.65	0.10	0.05	0.20	0.25

Ordem do schedule
[2,1]
- 4) Tarefas candidatas: 3, 4, 6
Verifica dentre tarefas candidatas a de menor chave aleatória: tarefa 6
Seleciona a tarefa prioritária e remove ela da lista

3	4	5	6	7	8	9	10	11	12
0.70	0.55	0.35	0.15	0.90	0.65	0.10	0.05	0.20	0.25

Ordem do schedule
[2,1,6]
- 5) Tarefas candidatas: 3, 4, 7
Verifica dentre tarefas candidatas a de menor chave aleatória: tarefa 4
Seleciona a tarefa prioritária e remove ela da lista

3	4	5	7	8	9	10	11	12
0.70	0.55	0.35	0.90	0.65	0.10	0.05	0.20	0.25

Ordem do schedule
[2,1,6,4]

Figura 6 – Representação da solução baseada no valor das chaves aleatórias.

Algoritmo	Meta-heurística BRKGA aplicada ao MSRCPS
Entradas:	$p, p_e, p_o, \rho_m, t_{limite}, n_{itermelhoria}$
Saída:	Melhor solução encontrada sol^*
1:	Inicializa o contador de iterações sem melhoria da solução da população vigente sol^+ :
2:	$cont_{itermelhoria} \leftarrow 0$
3:	Gere randomicamente Pop com p vetores de chaves aleatórias
4:	Decodifique e ordene Pop pelo valor da função objetivo
5:	Memorize a melhor solução encontrada: sol e sol^*
6:	Enquanto $t < t_{limite}$ faça:
7:	Crie o conjunto elite P_e^+ com base em p_e
8:	Crie o conjunto de filhos P_o^+ com base em p_e, ρ_m e $(p - p_e)$
9:	$Pop^+ \leftarrow P_e^+ \cup P_o^+$
10:	$Pop \leftarrow Pop^+$
11:	Decodifique e ordene Pop pelo valor da função objetivo
12:	Memorize a melhor solução encontrada sol^+
13:	Se $f(sol^+) < f(sol)$ ou $cont_{itermelhoria} > n_{itermelhoria}$ então:
14:	$sol^+ \leftarrow BuscaLocal(sol^+)$
15:	$sol \leftarrow sol^+$
16:	$cont_{itermelhoria} \leftarrow 0$
17:	Se $f(sol^+) < f(sol^*)$ então:
18:	$sol^* \leftarrow sol^+$
19:	$iter_{melhor} \leftarrow cont_{iter}$
20:	Fim Se
21:	Senão:
22:	$cont_{itermelhoria} += 1$
23:	Fim Se
24:	Se $cont_{iter} - iter_{melhor} > 3 \times n_{itermelhoria}$ então:
25:	$P_e^+ \leftarrow Perturbar(P_e^+)$
26:	Gere aleatoriamente $(p - p_e)$ novos indivíduos: P_m^+
27:	$Pop^+ \leftarrow P_e^+ \cup P_m^+$
28:	$Pop \leftarrow Pop^+$
29:	$iter_{melhor} \leftarrow cont_{iter}$
30:	Decodifique e ordene Pop pelo valor da função objetivo
31:	Fim Se
32:	Fim Enquanto
33:	Retorna sol^*

Algoritmo 1 – Pseudo-código do algoritmo desenvolvido.

Considerando-se o BRKGA clássico, o algoritmo desenvolvido baseado na referida meta-heurística encontra-se detalhado por meio do pseudo-código do Algoritmo 1. No Algoritmo em questão, destacam-se algumas diferenças do BRKGA clássico, a saber:

- Ausência de uma partição mutante na população. Sabendo-se que os referidos indivíduos são elementos usualmente de baixa qualidade e introduzidos com vistas à diversidade da população, optou-se pela inserção de genes mutantes, ou seja, aleatoriamente gerados, quando do cruzamento parametrizado uniforme. Dessa forma, com base em um parâmetro adicional, de possibilidade de inserção de gene mutante, p_m , para cada gene do cromossomo filho gera-se um número aleatório e se esse valor for inferior ao parâmetro anterior, o gene em questão receberá um novo valor aleatório; caso contrário, procede-se ao cruzamento parametrizado uniforme padrão. A Figura 7 ilustra o processo em comento;

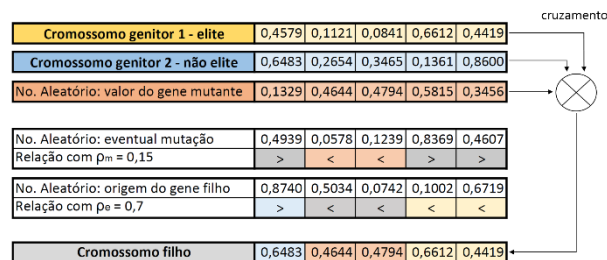


Figura 7 – Exemplo de geração de novo cromossomo no BRKGA adaptado.

- Adição de funções de busca local nas chaves aleatórias, aplicável em caso de melhoria de solução ou após um determinado número de iterações sem melhoria da solução. O procedimento consiste em se realizar alterações na chave aleatória ao invés de se alterar a solução já decodificada, de modo que as funções de busca local propostas são aplicáveis à representação de chaves aleatórias independentemente do decodificador ou do problema-alvo do algoritmo. No caso em questão, foram empregadas as seguintes funções de busca local:
 - *swap random key*: consiste na troca de todos os pares de genes possíveis do melhor indivíduo; e
 - *change resource key*: consiste na troca do valor da chave aleatória correspondente à regra de prioridade que define a ordem do emprego dos recursos. Altera-se o referido valor da chave aleatória para que se empregue todas as regras de prioridade definidas com base nas características dos recursos do problema-instância.
- Adição de funções de perturbação ou *shaking* na sequência do processo iterativo de geração de população do BRKGA, após um determinado número de iterações sem melhoria da solução. Nesse processo, cujo objetivo é diversificar a população vigente após um período sem melhoria da solução, realiza-se a geração de uma nova partição não-elite, acompanhada da realização de perturbações na população elite. Dessa forma, para cada indivíduo da elite, a depender de número sorteado de iterações, realiza-se a alteração de genes também sorteados desse cromossomo, por meio do emprego das seguintes funções de *shaking*:
 - *swap random key*: nessa função semelhante à empregada na busca local, sorteiam-se dois genes do cromossomo e troca-se o valor das respectivas chaves aleatórias;
 - *invert value*: sorteia-se um gene e inverte-se o valor de sua chave aleatória, ou seja, substitui o valor da chave aleatória rk pelo valor $1 - rk$;
 - *change random value*: o gene do cromossomo sorteado recebe novo valor aleatoriamente gerado;

- *swap neighbor*: após sortear um gene do cromossomo, troca o valor desse gene com o do vizinho à sua direita.

4. Resultados computacionais

Para a realização dos testes computacionais e comparativo de desempenho com resultados publicados da literatura, foram utilizadas as instâncias de [Myszkowski et al. 2015]. A seleção desse banco de instâncias foi realizada com base em características comuns entre o banco de instâncias e o problema real sob estudo: quantidade de tarefas ($n = 100$), de recursos ($k \in [5, 20]$) e de recurso-habilidade para sua execução ($r_{jt} = 1$).

Cada uma das instâncias foi resolvida pelo algoritmo em 20 execuções, em computador com processador Intel Core i7-9750H, 2.60 GHz, 8 GB de memória RAM. Para tal, realizou-se previamente a configuração de parâmetros do BRKGA modificado, utilizando-se da ferramenta computacional *iRace* de [López-Ibanez et al. 2016]. Nessa etapa de calibração, o *iRace* foi configurado para testar a melhor combinação de parâmetros expressos na Tabela 2, adaptados com base nos valores praticados em [Chaves e Lorena 2021], cujo resultado da calibração encontra-se expresso também na referida Tabela.

Parâmetro	Valores de entrada para o <i>iRace</i>	Melhor valor encontrado
p	{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000}	500
p_e	{0.1, 0.15, 0.20, 0.25, 0.30, 0.35}	0.2
ρ_e	{0.55, 0.60, 0.65, 0.70, 0.75, 0.80}	0.70
ρ_m	{0.01, 0.02, 0.03, 0.04, 0.05}	0.01
$n_{iter_melhoria}$	{100, 150, 200, 250, 300, 350, 400, 450, 500}	350

Tabela 2 – Parâmetros adotados no Algoritmo 1, baseados em [Chaves e Lorena 2021], após calibração realizada por meio do *iRace* de [López-Ibanez et al. 2016].

Os resultados consolidados por instância encontram-se na Tabela 3, bem como na Figura 8, que apresenta a distribuição de todas as soluções encontradas nas execuções realizadas. Como demonstrado, o algoritmo desenvolvido alcançou resultados equivalentes ao estado da arte da literatura em 61% das instâncias testadas, as quais foram selecionadas devido à similaridade com instâncias do problema real sob estudo. Em todas as instâncias testadas, a diferença percentual da melhor solução obtida foi inferior a 1%, tendo sido no pior caso 0,83% distante do melhor resultado de literatura para a instância. Ademais, as melhores soluções obtidas em cada instância apresentaram *gap*% médio de 0,27% com relação aos melhores resultados da literatura.

Instância		Algoritmos da literatura					Resultados do algoritmo desenvolvido			
Nome	BKS	GRASP	DEGR	GP-HH	M-PSO	ATMAS	Melhor sol.	Desv. Padr.	GAP%	tempo (s)
100_5_20_9_D3	387	401	392	387	-	392	387	0,00	0,00%	66.81
100_5_22_15	484	503	484	484	485	484	484	0,00	0,00%	40.87
100_5_46_15	528	552	529	528	530	529	528	0,00	0,00%	168.69
100_5_48_9	490	510	490	490	490	491	490	0,00	0,00%	253.94
100_5_64_9	471	494	474	474	471	475	474	0,30	0,64%	3604.11
100_5_64_15	478	501	481	481	478	482	481	0,00	0,63%	3602.67
100_10_26_15	233	251	234	233	233	237	233	0,47	0,00%	563.45
100_10_27_9_D2	207	221	215	207	-	216	207	0,62	0,00%	1382.86
100_10_47_9	252	263	255	252	257	257	252	0,30	0,00%	3554.76
100_10_48_15	243	256	244	243	244	245	243	0,46	0,00%	1241.28
100_10_64_9	240	255	243	241	240	245	242	0,85	0,83%	3607.93
100_10_65_15	243	256	244	243	244	247	244	0,48	0,41%	3605.34
100_20_22_15	126	134	130	126	128	136	127	0,57	0,79%	3613.51
100_20_23_9_D1	172	172	172	172	-	174	172	0,00	0,00%	6.79
100_20_46_15	161	170	164	161	162	164	161	0,00	0,00%	7.82
100_20_47_9	123	133	125	123	124	132	124	0,89	0,81%	3615.99
100_20_65_9	123	135	126	123	125	140	124	1,19	0,81%	3614.01
100_20_65_15	205	213	240	205	228	240	205	0,00	0,00%	17.65

Tabela 3 – Resultados obtidos com as instâncias de [Myszkowski et al. 2015], $n = 100$.

Comparando-se o desempenho do algoritmo desenvolvido com outros códigos também já elaborados para esse fim, denota-se que o BRKGA adaptado superou ou igualou o desempenho de algoritmos da literatura, em 72% das instâncias quando comparada com o GP-HH [Lin et al. 2020], em 83% do M-PSO [Quoc et al. 2022], bem como em 100% das instâncias quando comparado ao GRASP [Myszkowski e Siemieniński 2016], DEGR [Myszkowski et al. 2018] e ATMAS [Jedrzejowicz e Ratajczak-Ropel 2022]. Quanto à distribuição de todas as 360 (trezentos e sessenta) execuções realizadas, verifica-se que o resultado mediano obtido apresentou *gap%* de 0,41%, variando no intervalo de 0,00% a 0,85%. Levando-se em conta todas as execuções realizadas, em 79% dos testes o *gap%* foi inferior a 1%, enquanto em 89% dos ensaios realizados o *gap%* obtido foi igual ou menor a 2%. Os testes realizados demonstram a capacidade da meta-heurística convergir para boas soluções, obtendo os melhores resultados de literatura na maioria das instâncias testadas e apresentando competitividade frente a outros algoritmos de referência.

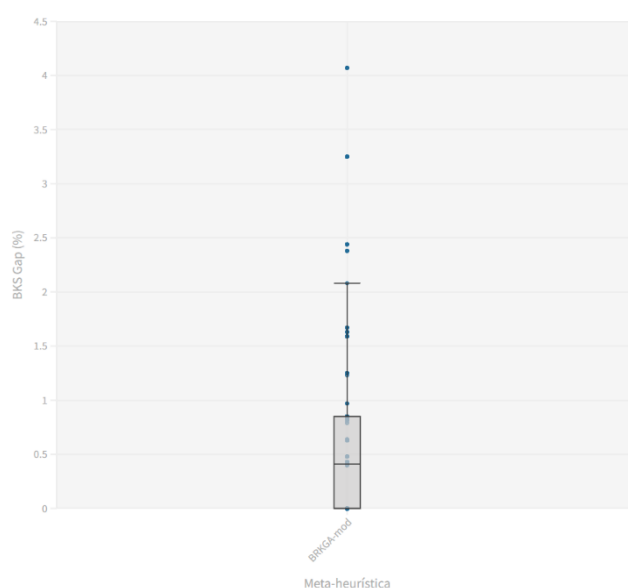


Figura 8 – Distribuição das soluções encontradas em todas as execuções do algoritmo.

5. Considerações finais

O presente artigo apresentou a problemática dos problemas de programação de projetos aplicáveis a escritórios de engenharia na área de construção civil. Para sua resolução, descreveu-se a modelagem matemática do MS-RCPSP, bem como foi proposto algoritmo baseado na meta-heurística BRKGA. Foram realizados testes computacionais com o algoritmo desenvolvido utilizando-se de instâncias da literatura e os resultados encontrados foram comparados com outros códigos elaborados para esse fim. Diante dos resultados encontrados, o BRKGA adaptado demonstrou a sua capacidade de resolução do problema em análise, igualando os melhores resultados da literatura parcialmente e superando três algoritmos de referência. Desse modo, fica evidenciado que o algoritmo em questão pode ser aplicado em instâncias reais ou baseadas em dados reais da problemática sob estudo, no processo de planejamento de tarefas de *portfolio* de projetos de engenharia e arquitetura, o que será proposto em continuidade da presente pesquisa, como trabalho futuro.

Agradecimentos

O presente trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processos 2018/15417-8 e 2016/01860-1, do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), processos 423694/2018-9 e 303736/2018-6, bem como da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), código de financiamento 001.

Referências

- Almeida, B. F.; Correia, I.; Saldanha-da-Gama, F. (2016). Priority-based heuristics for the multi-skill resource constrained project scheduling problem. *Expert Systems with Applications*, 57: 91–103.
- Almeida, B. F.; Correia, I.; Saldanha-da-Gama, F. (2018). A biased random-key genetic algorithm for the project scheduling problem with flexible resources. *Spanish Society of Statistics and Operations Research*, 26: 283–308.
- Baker, K. R.; Trietsch, D. (2018). Principles of sequencing and scheduling: Second edition. Wiley, Hoboken.
- Chaves, A. A. e Lorena, L. H. N. (2021). An Adaptive and Near Parameter-Free BRKGA Using Q-Learning Method. In *2021 IEEE Congress on Evolutionary Computation*, p. 2331-2338, Cracóvia, Polônia.
- Correia, I.; Lourenço, L. L.; Saldanha-da-Gama, F. Project scheduling with flexible resources: Formulation and inequalities. *OR Spectrum*, 34: 635–663.
- Gonçalves, J. F.; Mendes, J. J. M.; Resende, M. G. C. (2008). A genetic algorithm for the resource constrained multi-project scheduling problem. *European Journal of Operational Research*, 189: 1171–1190.
- Gonçalves, J. F.; Resende, M. G. C. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17: 487–525.
- Hamilton, A. (2008). Project design: Tasks that need to be managed. *Proceedings of Institution of Civil Engineers: Management, Procurement and Law*, 161: 133–134.
- Hartmann, S.; Briskorn, D. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 297:1-14.
- Jedrzejowicz, Piotr e Ratajczak-Ropel, Ewa (2022). A-Team Solving Multi-Skill Resource-Constrained Project Scheduling Problem. *Procedia Computer Science*, 207: 3294-3303.
- Laszczyk, M.; Myszkowski, P. B. (2019). Improved selection in evolutionary multi-objective optimization of Multi-Skill Resource-Constrained project scheduling problem. *Information Sciences*, 481: 412-431.
- Lin, Jian; Zhu, Lei Gao, Kaizhou (2020). A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. *Expert Systems With Applications*, 140.
- López-Ibáñez, M.; Dubois-Lacoste, J.; Cáceres, L. P.; Birattari, M.; Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3: 43–58.
- Montoya, C. et al. (2014). Branch-and-price approach for the multi-skill project scheduling problem. *Optimization Letters*, 8: 1721–1734.
- Myszkowski, P. B. et al. (2015). Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem. *Soft Computing*, 19: 3599-3619.
- Myszkowski P.B., Siemieniński J.J. (2016). GRASP applied to Multi-Skill Resource-Constrained Project Scheduling Problem. *Computational Collective Intelligence*, 9875: 402-411.
- Myszkowski P.B., Olech L.P., Laszczyk M. e Skowroński M.E. (2018). Hybrid Differential Evolution and Greedy (DEGR) for Solving Multi-Skill Resource-Constrained Project Scheduling Problem. *Applied Soft Computing*, 62: 1-14.
- Quoc, Huu Dang; The, Loc Nguyen e Doan, Cuong Nguyen (2022). An Effective Hybrid Algorithm Based on Particle Swarm Optimization with Migration Method for Solving the Multiskill Resource-Constrained Project Scheduling Problem. *Applied Computational Intelligence and Soft Computing*, 2022: 1-12.
- Snaauwaert, J.; Vanhoucke, M. (2021), A new algorithm for resource-constrained project scheduling with breadth and depth of skills. *European Journal of Operational Research*, 292:43-59.
- Snaauwaert, J.; Vanhoucke, M. (2023), A classification and new benchmark instances for the multi-skilled resource-constrained project scheduling problem. *European Journal of Operational Research*, 307:1-19.
- Spears, W.M.; Dejong, K.A. (1991). On the virtues of parameterized uniform crossover. In: *Proceedings of the Fourth International Conference on Genetic Algorithms*, 230–236.
- Young, K. D.; Feydy, T.; Schutt, A. (2017). Constraint programming applied to the multi-skill project scheduling problem. *Lecture Notes in Computer Science*, 10416: 308-317.