

DIFFERENTIAL EVOLUTION FOR GRAPH COLORING PROBLEM

Israel Pereira de Souza

Instituto de Computação - Universidade Federal Fluminense
Av. Gal. Milton Tavares de Souza, s/n, Boa Viagem – Niterói - RJ, 24210-346
israelsouza@id.uff.br

Igor Machado Coelho

Instituto de Computação - Universidade Federal Fluminense
Av. Gal. Milton Tavares de Souza, s/n, Boa Viagem – Niterói - RJ, 24210-346
imcoelho@ic.uff.br

Luiz Satoru Ochi

Instituto de Computação - Universidade Federal Fluminense
Av. Gal. Milton Tavares de Souza, s/n, Boa Viagem – Niterói - RJ, 24210-346
satoru@ic.uff.br

RESUMO

O Problema de Coloração de Grafos (do inglês, *Graph Coloring Problem (GCP)*) é um problema difícil conhecido na literatura. Em resumo, ele consiste em colorir um grafo usando o número mínimo de cores, de forma que não exista vértices adjacentes com a mesma cor. A literatura possui diversas aplicações para o GCP. A Evolução Diferencial é um algoritmo populacional evolucionário originalmente proposto para resolver problemas contínuos de otimização. Este trabalho apresenta um estudo de uma adaptação da Evolução Diferencial para o GCP. O algoritmo, chamado DE-GCP, foi capaz de encontrar o número cromático (solução ótima com o número mínimo de cores) em todas as 11 instâncias da literatura usadas nos experimentos, apresentando resultados competitivos com métodos da literatura.

PALAVRAS CHAVE. Número cromático, Problema de Coloração de Grafos, Evolução Diferencial.

Teoria e Algoritmos em Grafos, Metaheurísticas

ABSTRACT

Graph coloring problem (GCP) is a well-known hard-to-solve problem in literature. In short, it consists of coloring a graph using the minimum number of colors such that no adjacent vertices have the same colors. The literature presents several applications to GCP. Differential Evolution (DE) is a populational-based evolutionary algorithm originally proposed for solving continuous optimization problems. This work presents a study of an adaptation of DE to GCP. The algorithm, called DE-GCP is compared with literature algorithms using well-known instances benchmark. DE-GCP was able to find the chromatic number (optimum solution with the minimum number of colors) for all 11 instances in the literature used in the experiments, presenting competitive results with other literature methods.

KEYWORDS. Chromatic number. Graph Coloring Problem. Differential Evolution.

Graph Theory and Algorithms. Metaheuristics.

1. Introduction

Graph coloring problem (GCP) is a well-studied combinatorial optimization problem. Given an undirected graph $G = (V, E)$, the problem consists of coloring the vertices with a minimum number of colors (the chromatic number) such that no adjacent vertices get the same color. According to [Salari and Eshghi, 2005], the Graph coloring problem is expected to have a wide variety of applications such as scheduling, frequency assignment in cellular networks, timetabling, crew assignment, and other applications.

Differential Evolution (DE) is a population-based evolutionary algorithm originally proposed for solving continuous optimization problems [Storn and Price, 1997]. Solutions to a problem solved by DE are treated as individuals or chromosomes which are organized into populations. For every DE iteration, each chromosome in the population is designated as a target and DE employs mutation, crossover, and selection to create new candidate solutions. This work presents a DE algorithm study applied to the GCP.

Although DE was devised mainly for real parameter optimization, over the past few years, some works have been using DE to deal with discrete values. Two different methods can be highlighted in the literature review. Encoding methods are based on the application of some transformation scheme and a discrete method, named Discrete Differential Evolution (DDE) [Pan et al., 2008]. DDE implements mutation and crossover operators designed to be applied directly over the discrete solution representation such that the algorithm can directly work in the discrete domain [Wang et al., 2010].

According to [Rivera-López et al., 2020], two encoding methods have been utilized to represent the candidate solutions of combinatorial optimization problems: real-valued vectors and arrays of countables (binary or integer values). Still according to [Rivera-López et al., 2020], when the candidate solutions are encoded with countable values representing permutations, specialized variation operators such as the permutation matrix [Price et al., 2006], the adjacency matrix [Price et al., 2006], the list of movements scheme [Prado et al., 2010], the geometric search operators [Moraglio et al., 2013], and the algebraic-based scheme [Santucci et al., 2016] are needed to generate feasible solutions.

Recently, considering the multi-skill resource-constrained project scheduling problem, [Myszkowski et al., 2018] proposed hybrid Differential Evolution and Greedy Algorithm with a specialized indirect representation and transformation of solution space from discrete (typical for the problem), to continuous (typical for DE-approaches). [Zhou et al., 2018] studied a reentrant hybrid flow shop problem scheduling problem and presented, among other results, a hybrid differential evolution algorithm with an estimation of distribution algorithm using an ensemble model. They used two continuous-discrete transformation techniques known as forward and backward transformation proposed in [Onwubolu and Davendra, 2006].

Algorithms associated with discrete differential evolution (DDE) were proposed in [Pan et al., 2008; Wang et al., 2010; Tasgetiren et al., 2013; Hameed et al., 2020; Souza et al., 2023]. [Pan et al., 2008] proposed a novel discrete differential evolution algorithm for permutation flow shop scheduling problems. In the DDE they proposed, solutions are based on discrete item permutations. A mutant individual is obtained by perturbing the previous best solution of generation in the target population so that the differential variation is stochastically achieved through perturbation of the best solution from the previous generation in the target population. They also pointed out the characteristics that make their proposal different from the traditional GA in terms of solution-generation schemes. For additional related literature with discrete and combinatorial applications of DE algorithms, the reader is referred to [Das and Suganthan, 2011] and [Das et al., 2016].

The contributions of this paper are the study of a Differential Evolution algorithm with

a simple adaptation for combinatorial problems applied to the Graph Coloring Problem and the results comparison with literature methods, using well-known literature instances benchmark.

The rest of the article is organized as follows. Section 2 presents the description of the problem studied in this work. Section 3 describes the Differential Evolution in its original version presented by Rainer Storn and Kenneth Price [Storn and Price, 1997]. The DE algorithm for the GCP, called DE-GCP, is presented in Section 4. Section 5 discusses and analyzes the computational experiments of DE-GCP and two literature methods over 11 well-known instances benchmark from the literature. Section 6 concludes this work.

Nomenclatures used in this work are presented in Table 1.

Nomenclature	Description
DE	Differential Evolution
GCP	Graph Coloring Problem
$\chi(G)$	graph G chromatic number
q	solution number of colors
g	generation identifier
P_g	population P of generation g
NP	population size
F	mutation factor
CR	crossover ratio
$MaxGen$	maximum number of generations
$Stop_criteria$	algorithm stop criteria
$rand()$	randomly uniformed generated number $\in [0.0, 1.0]$
$X_{i,g}$	individual i in population g
$V_{i,g+1}, U_{i,g+1}$	temporary individuals generated from an individual $X_{i,g}$ in the evolutionary process of generation $g + 1$

Table 1: Nomenclatures.

2. Problem definition

The following definition was presented in [Salari and Eshghi, 2005]: “Let $G = (V, E)$ be an undirected graph where V is the set of vertices and E is the set of edges. An independent set is a subset of vertices in which no pair of adjacent vertices exists. A q -coloring of G is a mapping $c : V \rightarrow \{1, 2, 3, \dots, q\}$ that assigns colors to vertices. The coloring is feasible if no two adjacent vertices have the same color, i.e. $\forall u, v \in E : c(u) \neq c(v)$, otherwise, conflicts happen. A coloring with at least one conflict is called an infeasible coloring. An optimal coloring of G is a feasible coloring with the smallest number of colors. This minimum number of colors q for which a feasible q -coloring exists is called the chromatic number and is denoted by $\chi(G)$. Given a graph G , the graph coloring problem is to find an optimal coloring. In addition, graph coloring can be formulated as a set- partitioning problem; a feasible q -coloring is a partitioning of set V into q independent sets, i.e equation (1). Independent sets are also called color classes. By analogy, the objective is to partition the set V into a minimum number of color classes.”

$$V = \bigcup_{i=1}^q C_i : C_i \cap C_j = \emptyset, \quad \forall i, j, i \neq j. \quad (1)$$

Figure 1 presents an example of a graph coloring with ten vertices and three colors, also with a chromatic number of three.

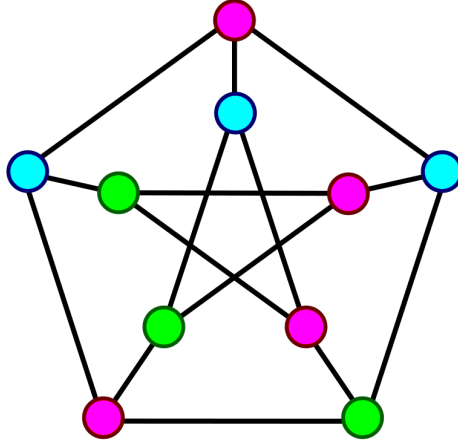


Figure 1: Example of a graph coloring with 10 vertices and 3 colors.

3. Differential Evolution

Differential evolution (DE) was proposed by Storn and Price in 1997 for minimizing non-differentiable and nonlinear continuous functions [Storn and Price, 1997]. DE is a population-based metaheuristic that optimizes a problem by iteratively improving candidate solutions in an evolutionary process. Each DE iteration creates a new generation of chromosomes called a population.

To create a new DE generation, every chromosome (also called vector or individual) in the current population is determined as a target, and a trial chromosome is created using mutation and crossover operators. To determine which chromosome will be in the next generation the selection method is applied with a greedy criterion. If the trial achieves a better value than the target chromosome, it is selected, otherwise, the target is used again.

DE has three control parameters, which are the population size NP , the mutation factor F (also called differential weight or scaling factor), and the crossover ratio CR (also called crossover probability). In the rest of this section, we detail the main DE elements. The mutation is described in Section 3.1, the crossover is described in Section 3.2, the selection is described in Section 3.3, and DE strategies are described in Section 3.4.

3.1. Mutation

For each target $X_{i,g}$ from the population P_g , three other distinct chromosomes $X_{r_1^i,g}$, $X_{r_2^i,g}$, $X_{r_3^i,g}$ are randomly selected, with $i \neq r_1^i \neq r_2^i \neq r_3^i$. A mutant individual $V_{i,g}$ is created by the weighted difference of $X_{r_2^i,g}$ and $X_{r_3^i,g}$, with the amplification controlled by the mutation factor F , and added to $X_{r_1^i,g}$. The mutation operation is presented by Equation (2).

$$V_{i,g+1} = X_{r_1^i,g} + F \cdot (X_{r_2^i,g} - X_{r_3^i,g}), \quad F \in (0, 2] \quad (2)$$

3.2. Crossover

The crossover operator is applied to increase the diversity of generated chromosomes [Storn and Price, 1997]. A chromosome $U_{i,g+1}$ can be represented by a sequence of components $U_{i,g+1} = (U_{1i,g+1}, U_{2i,g+1}, \dots, U_{Di,g+1})$, where D is the number of components of the chromosome.

For each component j of the chromosome, a random number $rand(j)$ is generated, and if $rand(j) \leq CR$, then the j -component of the mutant takes place on the trial chromosome, otherwise the j -component of the target is used. To ensure that the trial will be different from the target chromosome, a random component of the mutant is taken without CR evaluation.

Equation 3 presents the crossover operation.

$$U_{ji,g+1} = \begin{cases} V_{ji,g+1}, & \text{if } rand(j) \leq CR, \quad CR \in [0, 1] \\ X_{ji,g}, & \text{otherwise.} \end{cases} \quad (3)$$

Figure 2 was obtained in [Storn and Price, 1997] and presents an crossover example.

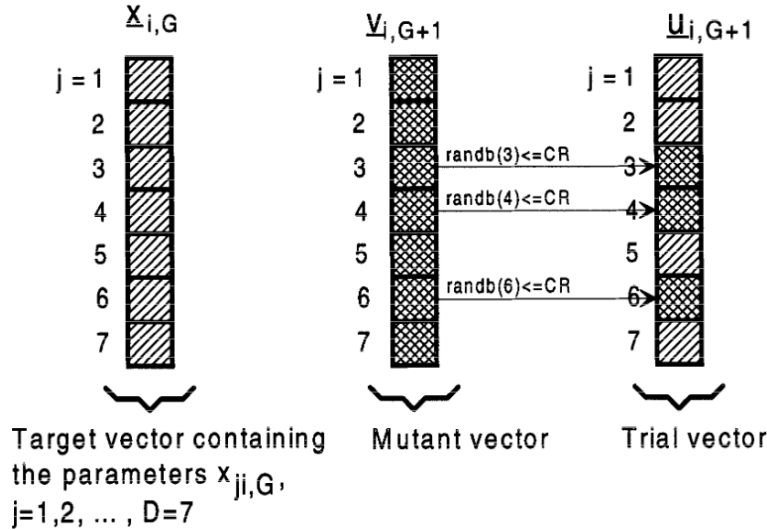


Figure 2: Crossover example (source:[Storn and Price, 1997]).

3.3. Selection

To keep the population size constant over subsequent generations, the selection is performed to determine whether the target ($X_{i,g}$) or the trial chromosome ($U_{i,g+1}$) survives to the next generation. Selection applies a greedy criteria over the target $X_{i,g}$ and the trial $U_{i,g+1}$. If $U_{i,g+1}$ achieves a lower cost compared to $X_{i,g}$, then $U_{i,g+1}$ takes its place on the generation $g + 1$. Otherwise, the target $X_{i,g}$ is kept, remaining in the next generation as $X_{i,g+1}$. Selection assures that population P_{g+1} is equally or better than the previous populations.

3.4. DE strategies

DE literature presents several strategies (also known as techniques or variants) for the design of the mutation and crossover operators. Classical DE strategies presented by [Storn and Price, 1997] are denoted as $DE/x/y/z$, where: x specifies the mutation type *rand* or *best*, which defines if the first selected individual in the mutation ($X_{r_1^i,g}$) will be a random or best individual of the population, respectively; y represents the number of weighted differences used on mutation, generally 1 or 2; and z specifies the crossover type (*bin* or *exp*).

The *bin* (binary) crossover type verifies all individual components for the operation while in the *exp* (exponential) crossover, the check is halted when the first randomly raffled value greater than *CR* is reached. A non-classical DE strategy example, *DE/rand-to-best/y/z* [Onwubolu and Davendra, 2009], is specified only for the mutation operator. It adds another weighted difference with the best individual on the random mutation type, with the purpose of combining the faster convergence of the *best* strategy with the robustness of the *rand* strategy. Sections 3.1 and 3.2 describe *DE/rand/1/bin* strategy.

4. DE for Graph Coloring Problem

In this section we present the DE algorithm for the GCP, called DE-GCP. The algorithm consists of minimizing the number of conflicts in a graph coloring. Once no conflict is obtained, the solution is viable and if it is the chromatic number, then the optimum solution is found.

Since GCP is a combinatorial problem, an adaptation for DE Mutation is necessary. Many Mutation encodings for DE were proposed in the literature and are briefly described in Section 1. For DE-GCP, rounding for the nearest integer was used. In addition, if a value outside range $[1, q]$ is obtained, then a random number $r \in [1, q]$ is used instead.

The algorithm has two optimization modes: the first has the number of colors defined (q , given as an input parameter) and the latter tries to find the graph chromatic number algorithmically.

When q is defined, all solutions are generated with q colors, once a solution with no conflict is obtained then the algorithm stops and presents the solution. In other words, the algorithm *Stop_criteria* consists of reaching the maximum number of generations (*MaxGen*) or reaching a feasible solution (a solution without conflicts). It is possible to not find a feasible solution in this version. When q is not defined (it means that q will be minimized algorithmically), the initial population starts with a feasible solution, where a greedy solution constructor is used to find a q number. Then, q is decreased by one and the algorithm tries to find a feasible solution. If a feasible solution is achieved, then q is reduced to one number and this process repeats. In addition, the algorithm *Stop_criteria* is defined as reaching the maximum number of generations (*MaxGen*) or reaching a feasible solution. A feasible solution is always found in this version.

Algorithm 1 describes the DE-GCP algorithm. The inputs are the graph G , F , CR , NP DE parameters, DE *Stop_criteria*, and optionally, the number of colors q to color the graph G . The algorithm starts initializing the generation identifier with 0. If q was not defined at the input phase, then q will be minimized algorithmically. In this case, a greedy constructor is generated its solution is defined as the best current solution X_{best} , and the number of colors used in X_{best} is defined as q . Otherwise, if q is defined in the input phase, this greedy constructor is skipped and the initial population is generated at line 5. The initial population randomly colors every population individual using colors $\in [1, q]$. Again, if q was not defined at the input phase, q is decreased at one unit and the algorithm tries to color the individuals with the new q number of colors (currently, one less color than current X_{best} solution). Then DE principal operators take action in the loop from lines 11-21. Crossover and Selection are applied according to Section 3. For Mutation, since GCP is a combinatorial problem, an adaptation for DE Mutation is necessary. For DE-GCP, rounding for the nearest integer was used. In addition, if a value outside range $[1, q]$ is obtained, then a random number $r \in [1, q]$ is used instead. Line 23 returns the best feasible solution.

Algorithm 1: DE-GCP pseudocode

Input: $G, F, CR, NP, Stop_criteria, q$.
 1 $g \leftarrow 0$
 2 **if** q will be minimized **then**
 3 $X_{best}, q \leftarrow \text{Greedy_constructor}(G)$
 4 **end if**
 5 $P_0 \leftarrow \text{Generate Initial Population}(G, NP, q)$
 6 **do**
 7 **if** q will be minimized **then**
 8 $q \leftarrow q - 1$
 9 **end if**
 10 **do**
 11 $P_{g+1} \leftarrow \emptyset$
 12 **for** $i = 1$ to NP **do**
 13 randomly select $X_{r_1, g}, X_{r_2, g}, X_{r_3, g} \in P_g, i \neq r_1 \neq r_2 \neq r_3$
 14 $V_{i, g+1} \leftarrow \text{Mutation}(X_{r_1, g}, X_{r_2, g}, X_{r_3, g}, F, q)$
 15 $U_{i, g+1} \leftarrow \text{Crossover}(X_{i, g}, V_{i, g+1}, CR)$
 16 $X_{i, g+1} \leftarrow \text{Selection}(X_{i, g}, U_{i, g+1})$
 17 $P_{g+1} \leftarrow P_{g+1} \cup X_{i, g+1}$
 18 **end for**
 19 $X_{best} \leftarrow \text{select a feasible solution from } P_{g+1} \text{ if it exists}$
 20 $g \leftarrow g + 1$
 21 **while** $Stop_criteria$ is not reached;
 22 **while** a feasible solution with a reduced q was obtained;
 23 **return** X_{best}

5. Computational experiments

In this section, we present and discuss the results of the computational experiments performed. A sequential version of the DE-GCP algorithm was coded in C language, compiled using *gcc*, version 11.4.0. The experiments were carried out on a personal computer with an AMD Ryzen 5 3600 processor with 16GB RAM, running under a Linux system Ubuntu 22.04.

An extensive DE parameter calibration was conducted in [Souza et al., 2023]. The most performing values for F, CR , and DE technique were selected for DE-GCP experiments and are presented in Table 2. The population size NP was defined using the value of the compared algorithms.

Parameter	Value
NP	100
F	0.7
CR	0.7
DE strategy	rand/1/exp
MaxGen	500

Table 2: DE-GCP parameter values for the experimental tests.

The experiments were conducted using well-known literature benchmarks. It can be ob-

tained AT. All instances considered in [Agrawal and Agrawal, 2015] were tested. The instances Expected $\chi(G)$ was obtained at [Agrawal and Agrawal, 2015].

DE-GCP results were compared with two metaheuristics: PSO [Agrawal and Agrawal, 2015] and APSO [Agrawal and Agrawal, 2015], which are both populational metaheuristics, similarly to DE. Table 3 presents the performance comparison of PSO, APSO, and DE-GCP. The first three columns present the instance name, its number of vertices, and edges. The fourth column presents the Expected graph chromatic number $\chi(G)$. The fifth and sixth columns present the PSO total number of steps and the number of color results. The seventh and eighth column presents the APSO total number of steps and the number of colors result. The ninth and tenth column presents the DE-GCP generation that found the solution and the number of color results. Results presented with $\emptyset$ means that the algorithm was unable to find the solution.

Instance	Vertices	Edges	Expected	PSO		APSO		DE-GCP	
			$\chi(G)$	Total steps	Results	Total steps	Results	Generation	Results
myciel3.col	11	20	4	14	4	1	4	1	4
myciel4.col	23	71	5	59	5	1	5	1	5
myciel5.col	47	236	6	100	$\emptyset$	1	6	1	6
huck.col	74	301	11	100	$\emptyset$	1	11	1	11
jean.col	80	254	10	100	$\emptyset$	1	10	1	10
david.col	87	406	11	100	$\emptyset$	1	11	1	11
games120.col	120	638	9	100	$\emptyset$	1	9	1	9
miles250.col	128	387	8	100	$\emptyset$	1	8	1	8
miles1000.col	128	3216	42	100	$\emptyset$	2	42	237	42
anna.col	138	493	11	100	$\emptyset$	1	11	1	11
queen5_5.col	25	160	5	100	$\emptyset$	32	5	22	5

Table 3: Algorithms performance comparison.

PSO found a solution with the chromatic number only for two instances. APSO and DELS-GCP were able to find the solution with chromatic numbers for all tested instances. DE-GCP was able to find the solution with a chromatic number for 9 of 11 instances with one generation. APSO was able to find the solution with a chromatic number for 9 of 11 instances with one total step. DE-GCP was able to find the solution for instance “queen5_5.col” in fewer generations than APSO total steps, but considering instance “miles1000.col”, APSO was able to find the solution in fewer total steps than DE-GCP generations. Overall, DE-GCP presented a competitive algorithm, outperforming PSO, where DE-GCP found the solution with a chromatic number on all 11 instances, against PSO, that found in only 2 instances. Also, DE-GCP presented competitive results with APSO. DE-GCP found the solution for instance “queen5_5.col” in fewer generations than APSO total steps, but APSO outperformed DE-GCP in the same aspect for instance “miles1000.col”.

6. Conclusions

This work presented a Differential Evolution algorithm for the Graph Coloring Problem, called DE-GCP. DE-GCP was compared with two literature algorithms (PSO and APSO) which are both populational similar to DE. The experiments were conducted using all instances presented in [Agrawal and Agrawal, 2015] for comparison, which are 11 known instances in the literature of GCP. DE-GCP presented competitive results and was able to outperform PSO, finding the chromatic

number on all 11 instances, against the PSO which only found the chromatic number on 2 of these instances. DE-GCP was competitive with APSO, finding the chromatic number in fewer generations than APSO in one instance, however, the APSO was able to find the chromatic number in fewer generations against DE-GCP in one instance. Both algorithm DE-GCP and APSO were able to find the optimum on these 11 instances.

In future work, the application of other constructive algorithms as well as the application of local search methods can be studied. The hybridization study with other metaheuristics methods can also be applied. DE-GCP can be studied further with the application of more benchmark instances as well as the comparison with other literature methods.

References

- Jitendra Agrawal and Shikha Agrawal. Acceleration based particle swarm optimization for graph coloring problem. *Procedia Computer Science*, 60:714–721, 2015.
- Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, 2011. doi: 10.1109/TEVC.2010.2059031.
- Swagatam Das, Sankha Subhra Mullick, and P.N. Suganthan. Recent advances in differential evolution – an updated survey. *Swarm and Evolutionary Computation*, 27:1–30, 2016. ISSN 2210-6502. doi: <https://doi.org/10.1016/j.swevo.2016.01.004>.
- A Hameed, B Aboobaidar, M Mutar, and N Choon. A new hybrid approach based on discrete differential evolution algorithm to enhancement solutions of quadratic assignment problem. *International Journal of Industrial Engineering Computations*, 11(1):51–72, 2020.
- A. Moraglio, J. Togelius, and S. Silva. Geometric differential evolution for combinatorial and programs spaces. *Evolutionary Computation*, 21(4):591–624, 2013. doi: 10.1162/EVCO_a.00099.
- Paweł B. Myszkowski, Łukasz P. Olech, Maciej Laszczyk, and Marek E. Skowroński. Hybrid differential evolution and greedy algorithm (degr) for solving multi-skill resource-constrained project scheduling problem. *Applied Soft Computing*, 62:1–14, 2018. doi: <https://doi.org/10.1016/j.asoc.2017.10.014>.
- Godfrey Onwubolu and Donald Davendra. Scheduling flow shops using differential evolution algorithm. *European Journal of Operational Research*, 171(2):674–692, 2006. ISSN 0377-2217. doi: <https://doi.org/10.1016/j.ejor.2004.08.043>.
- Godfrey C Onwubolu and Donald Davendra. *Differential evolution: A handbook for global permutation-based combinatorial optimization*, volume 175. Springer Science & Business Media, 2009.
- Quan-Ke Pan, Mehmet Fatih Tasgetiren, and Yun-Chia Liang. A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers & Industrial Engineering*, 55(4):795–816, 2008. doi: <https://doi.org/10.1016/j.cie.2008.03.003>.
- Ricardo S. Prado, Rodrigo C. P. Silva, Frederico G. Guimarães, and Oriane M. Neto. Using differential evolution for combinatorial optimization: A general approach. In *2010 IEEE International*

- Conference on Systems, Man and Cybernetics*, pages 11–18, 2010. doi: 10.1109/ICSMC.2010.5642193.
- Kenneth Price, Rainer M Storn, and Jouni A Lampinen. *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.
- Rafael Rivera-López, Efrén Mezura-Montes, Juana Canul-Reich, and Marco Antonio Cruz-Chávez. A permutational-based differential evolution algorithm for feature subset selection. *Pattern Recognition Letters*, 133:86–93, 2020. doi: <https://doi.org/10.1016/j.patrec.2020.02.021>.
- Ehsan Salari and Kourosh Eshghi. An aco algorithm for graph coloring problem. In *2005 ICSC Congress on computational intelligence methods and applications*, pages 5–pp. IEEE, 2005.
- Valentino Santucci, Marco Baiocchi, and Alfredo Milani. Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion. *IEEE Transactions on Evolutionary Computation*, 20(5):682–694, 2016. doi: 10.1109/TEVC.2015.2507785.
- Israel Pereira Souza, Maria Claudia Silva Boeres, and Renato Elias Nunes Moraes. A robust algorithm based on differential evolution with local search for the capacitated vehicle routing problem. *Swarm and Evolutionary Computation*, 77:101245, 2023.
- Rainer Storn and Kenneth Price. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.
- M. Fatih Tasgetiren, Quan-Ke Pan, P. N. Suganthan, and Ikbale Ece Dizbay. Metaheuristic algorithms for the quadratic assignment problem. In *2013 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS)*, pages 131–137, 2013. doi: 10.1109/CIPLS.2013.6595210.
- Ling Wang, Quan-Ke Pan, P.N. Suganthan, Wen-Hong Wang, and Ya-Min Wang. A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. *Computers & Operations Research*, 37(3):509–520, 2010. doi: <https://doi.org/10.1016/j.cor.2008.12.004>.
- Bing-hai Zhou, Li-man Hu, and Zhen-yi Zhong. A hybrid differential evolution algorithm with estimation of distribution algorithm for reentrant hybrid flow shop scheduling problem. *Neural Computing and Applications*, 30(1):193–209, 2018. doi: <https://doi.org/10.1007/s00521-016-2692-y>.