A Hybrid Differential Evolution Algorithm – Game Theory for the Berth Allocation Problem

Nasser R. Sabar, Siang Yew Chong, and Graham Kendall

The University of Nottingham Malaysia Campus,
Jalan Broga, 43500 Semenyih, Selangor, Malaysia
{Nasser.Sabar,siang-yew.chong,Graham.Kendall}@nottingham.edu.my

Abstract. The berth allocation problem (BAP) is an important and challenging problem in the maritime transportation industry. BAP can be defined as the problem of assigning a berth position and service time to a given set of vessels while ensuring that all BAP constraints are respected. The goal is to minimize the total waiting time of all vessels. In this paper, we propose a differential evolution (DE) algorithm for the BAP. DE is a nature-inspired meta-heuristic that has been shown to be an effective method to addresses continuous optimization problems. It involves a population of solutions that undergo the process of selection and variation. In DE, the mutation operator is considered the main variation operator responsible for generating new solutions. Several mutation operators have been proposed and they have shown that different operators are more suitable for different problem instances and even different stages in the search process. In this paper, we propose an enhanced DE that utilizes several mutation operators and employs game theory to control the selection of mutation operators during the search process. The BAP benchmark instances that have been used by other researchers are used to assess the performance of the proposed algorithm. Our experimental results reveal that the proposed DE can obtain competitive results with less computational time compared to existing algorithms for all tested problem instances.

Keywords: Differential evolution, berth allocation problem, meta-heuristics, optimization.

1 Introduction

Maritime transportation has experienced a tremendous growth of container usage over the last two decades [1], [2]. Port managers face great challenges in providing effective and efficient services. The berth allocation problem (BAP) is one of the main challenges confronting port managers. Providing an efficient solution to the BAP plays an important role in improving port effectiveness [2]. BAP seeks to assign, for each vessel, a berth position and service time on the selected berth. The goal is to minimize the total waiting time of all vessels as far as possible [1].

BAP is an NP-hard problem [1]. Small instances can be solved optimality using exact methods. However, they become impractical as the size of the instances in-

H. Handa et al. (eds.), *Proc. of the 18th Asia Pacific Symp. on Intell. & Evol. Systems – Vol. 2,*
Proceedings in Adaptation, Learning and Optimization 2, DOI: 10.1007/978-3-319-13356-0_7

creases [2]. As such, researchers have utilized meta-heuristic algorithms to deal with large-scale instances as they can often provide good quality solutions within realistic computational times. Examples of meta-heuristic algorithms being utilized for BAP include: tabu search [1], clustering search [3] particle swarm optimization [4] and hybrid column generation approach [5].

In this paper, we propose a differential evolution (DE) algorithm for the BAP. DE is a nature-inspired population-based meta-heuristic that has been demonstrated to be efficient and effective for many hard continuous optimization problems. DE operates on a population of solutions and iteratively improves them. In each iterative step or generation, a new solution is generated using two variation operators: mutation and crossover. The mutation operator in DE is considered the primary variation operator and several mutation operators have been introduced. However, it is not known in advance which operator should be used as different operators work well for different problem instances with different characteristics and in different stages of the search process [6], [7], [8], [9]. In this paper, we propose an enhanced DE that utilizes a variation operator with multiple mutation operators for the BAP. We utilize game theory to provide a mechanism to control the selection of mutation operators throughout the search process of DE. The performance of the proposed algorithm has been assessed using the existing BAP benchmark instances [1]. The experimental results reveal that the proposed algorithm can obtain competitive results with less computational time when compared with existing algorithms.

2 Problem Description

Bierwirth and Meisel [2] has classified BAP into two types according to the berth type and the vessel's arrival time. The berth type is categorized as discrete if the quay has been partitioned into a set of berth sections and continuous if the quay is not partitioned. The vessel's arrival time is categorized as dynamic if the vessels can arrive at any time during the planning horizon, and static if all vessels have arrived at the port before the berth planning begins. We focus on the discrete dynamic BAP [2], [10]. For this BAP, there are a set of berth sections with predefined lengths and a set of vessels. Each vessel has an arrival time, priority, vessel length and handling time [2]. Some of them can be allocated to any berths based on vessel lengths while others can only be allocated to a subset of berths. The vessel handling time is different from one berth to another. The overall goal is to allocate for each vessel a berth section and service time (berthing time) on the allocated berth while respecting the following constraints [1]:

- Each vessel is allocated to exactly one berth.
- There is no more than one vessel allocated to the same berth at the same time (same service time).
- Each berth can handle at most one vessel at any given time.

The main role of the optimization algorithm is to minimize the total waiting time of all vessels which is calculated as follows (objective function) [1]:

$$\min \sum_{i \in n} \sum_{k \in m} v_i \left[T_i^k - a_i + + t_i^k \sum_{j \in n} x_{ij}^k \right] \quad (1)$$

where

- n : number of vessels
- m : number of berths
- v_i : the priority of vessel i
- T_i^k : the berting time of a vessel i at berth k.
- a_i : the arrival time of vessel i.
- t_i^k : the handling time of vessel i at berth k.
- x_{ij}^k : decision variable, $x_{ij}^k = 1$ if vessel j is serviced by berth k after the vessel i and $x_{ij}^k = 0$ otherwise.

3 Proposed Methodology

In this section, we first present the basic DE algorithm that is followed by the proposed approach to enhance DE for the BAP problem.

3.1 Basic Differential Evolution Algorithm

The DE algorithm was proposed in [11] to deal with continuous optimization problems (real-valued fitness functions). It belongs to a class of nature-inspired, population-based meta-heuristic algorithms [12]. A general DE algorithm starts with a population of solutions and then applies evolutionary operators of variation (mutation and crossover) and selection to improve the population of solutions iteratively over a certain number of generations. For every solution in the population, DE generates a new solution using the mutation operator that randomly selects three different solutions from the current population and combines them according to a prescribed operation. The new generated solution is then combined with the parent solution using the crossover operator to generate an offspring. The selection step first calculates the fitness of the offspring and then replaces it with the parent solution if it has a better fitness. This process is repeated for a predefined number of generations.

Over the years, many DE variants that use different mutation or crossover operators have been proposed. A general DE scheme use the notation *DE/x/y/z*, where *x*

represents the base solution to be mutated, *y* defines the number of different solutions to be used to perturb *x*, and *z* denotes the crossover type, *binomial* or *exponential* [11], [12]. A well-known DE variant is the "*DE/rand/*1*/bin*", where "*DE*" is Differential Evolution, "*rand*" means the solutions will be randomly selected, "1" indicates the number of pairs of solutions and "*bin*" indicates that binomial crossover will be used. The basic steps of the *DE/rand/*1*/bin* are as follows [11]:

***Step 1*:** Randomly generate a population of solutions, *NP*.
***Step 2*:** Calculate the fitness, *f*, of the population.
***Step 3*:** For each parent solution (x_i^G) in the current population *NP* (*i* is the solution index and *G* is the current generation) generate a new solution (m_i^G) using (2):

$$m_{i,j}^G = x_{1,j}^G + F * (x_{2,j}^G - x_{3,j}^G), \forall j \in \{1,...,n\} \tag{2}$$

where *j* represents the index of current decision variable, *n* is the maximum number of decision variables in a given problem instance, *F* is the scaling factor ($F \in [0, 1]$) and x_1^G , x_2^G and x_3^G are three randomly chosen solutions from the current population where $x_1^G \neq x_2^G \neq x_3^G$.

***Step 4*:** Apply the crossover operator to combine the solution (m_i^G) generated by the mutation operator with the parent solution (x_i^G) based on the crossover rate *CR* ($CR \in [0, 1]$) in order to generate a new offspring (m_i^{G+1}) as follows (3):

$$m_{i,j}^{G+1} = \begin{cases} m_{i,j}^G & if\ Rand(j) \leq CR\ or\ j = Rnd(i) \\ x_{i,j}^G & if\ Rand(j) > CR\ and\ j \neq Rnd(i) \end{cases} \tag{3}$$
$$\forall j \in \{1,...,n\}, \forall i \in \{1,...,|NP|\}$$

where *Rand(j)* is a random number (*Rand(j)* ∈ [0, 1]) selected for the j^{th} decision variable, *Rnd(i)* is a random decision variable index (*Rnd(i)* ∈ {1,…, *n*}). *Rnd* ensure that m_i^{G+1} gets at least one decision variable from m_i^G .

***Step 5*:** Calculate the fitness of m_i^{G+1} and compare it with x_i^G . Replace x_i^G with m_i^{G+1} if m_i^{G+1} fitness is better than x_i^G as follows (4):

$$x_i^{G+1} = \begin{cases} m_i^{G+1} & if\ f(m_i^{G+1}) \leq f(x_i^G) \\ x_i^G & if\ f(m_i^{G+1}) > f(x_i^G) \end{cases} \tag{4}$$
$$\forall i \in \{1,...,|NP|\}$$

Step 6*:** If the termination criterion is satisfied (the number of generations), stop and return the best solution. Otherwise, go to ***Step 3.

4 The Proposed Algorithm

The variety of problem instances having different characteristics, or those with complex structures (e.g., different sections having different structures), makes it challenging to know in advance the best variation operator to use (i.e., blackbox optimization) [12]. Each one has its own strength and weakness and may work well for certain instances or at a certain stages in the search process. Consequently, several DE frameworks that utilize a set of mutation operators have been proposed [6], [7], [8], [9]. These frameworks seek to combine the strength of several mutation operators in one framework that can effectively solve the given problem instances [13], [14], [15]. In this paper, we propose a DE algorithm that utilizes several mutation operators to solve the BAP. The proposed DE makes use of game theory in the design of the selection mechanism that chooses the operator to be used at the current search stage. In the following, we first describe the solution representation and the population generation method, and later the utilized mutation operators and proposed selection mechanism.

4.1 Solution Representation and the Population Generation Method

DE was originally proposed to solve continuous optimization problems [12]. To deal with combinatorial optimization problems such as BAP, a suitable solution representation or a decoding scheme is needed to convert the real numbers into integers [12]. In BAP, each vessel has to be assigned to a berth section and being given a service time on the selected berth. This implies that we need to deal with the assignment problem that is responsible for assigning for each vessel a berth section and the scheduling problem that assigns a service time for each vessel. In this paper, we avoid modifying the DE mutation operator by having a decimal representation of BAP solutions $a_0.a_1a_2$...where the integer part a_0 represents assigned berth sections while the fraction parts a_1a_2... represent the order of this vessel on this berth [16]. Figure 1 shows an example of BAP solution representation. Here, an instance of BAP has 6 vessels (n=6) that are needed to be assigned to 3 berths. If we assign numbers for the vessels from 1 to 6, the decision variables will be (1, 2, 3, 4, 5, 6), as shown in the first row of Table 1. Next, we generate for each decision variable a random number r ($r \in [1, 3]$)), as shown in second row of Table 1.

Table 1 can be decoded into a BAP solution as follows: vessel 1 is assigned to the second berth and is second in order on this berth, vessel 2 is assigned to the first berth and is first in order on this berth, vessel 3 is assigned to the first berth and is second in order on this berth, and so on. Next, on each berth we sort the assigned vessels in an ascending order based on their arrival time. In this paper, the initial population of solutions of DE is randomly generated by assigning each decision variable a random value between 1 and the maximum number of vessels in a given problem instance. The generated solutions are assigned fitness values using equation (1).

Table 1. BAP solution representation

Vessel index	*1*	*2*	*3*	*4*	*5*	*6*
Decision variables	*2.2*	*1.1*	*1.2*	*3.2*	*3.1*	*2.1*

4.2 Mutation Operators and the Selection Mechanism

In this paper, the proposed DE makes use of the following mutation operators [11], [12]:

- M_1: *DE/rand/1/bin*, $m_i = x_1 + F * (x_2 - x_3)$
- M_2: *DE /best /1/bin*, $m_i = x_{best} + F * (x_1 - x_2)$
- M_3: *DE* /parent-to- best/1, $m_i = x_i + F * (x_{best} - x_i) + F * (x_1 - x_2)$
- M_4: *DE /best /2/ bin*, $m_i = x_{best} + F * (x_1 - x_2) + F * (x_3 - x_4)$

where "*parent*" (x_i) is the parent solution to be perturbed, "*best*" indicates the best solution in the population and x_1, x_2, x_3 and x_4 are randomly selected solutions from the current population where $x_1 \neq x_2 \neq x_3 \neq x_4 \neq x_i \neq x_{best}$

In this paper, each parent solution is associated with a set of mutation operators and, at each generation, one of them is selected to generate a new solution. We employ a game theoretic concept to design the selection mechanism of the mutation operators for each parent. Each strategy is associated with a profile that keeps the history of the strategy performance. As in [17], we model the selection mechanism in the context of a two-player game. The DE population of solutions is divided into two to play two-player game. Each solution represents a player with a set of mutation operators (M_1, M_2, M_3 and M_4) representing strategies that can be played. Each strategy will be assigned a payoff representing the improvement obtained by the selected strategy. According to the strategy probability distribution, each player selects one strategy to play against another strategy. Based on the obtained result, the player will update the strategy profile and the payoff. In this paper, the strategy profile keeps the accumulated payoff of each one and it is updated at every generation. The payoff of each strategy is calculated as follows: let S [] be the array of the probability of selecting the strategy, f_p and f_n represents the fitness values of the parent and generated solutions, NS represents the number of strategies. Then, if the application of the i-th strategy improves the fitness value of the parent solution, the payoff of the i-th strategy is updated as follows: $S[i]=S[i]+\Delta$ where $\Delta=(f_p - f_n)/ (f_p + f_n)$, $\forall\ j \in \{1,\ldots,NS\}$ and $j \neq i$, $S[j]=S[j]-(\Delta/(NS-1))$. Otherwise (if the solution cannot be improved), $S[i]=S[i]-|(\Delta*\alpha)|$ where $\alpha=Current\_Genetraion/Total\_Generations$, $\forall\ j \in \{1,\ldots,NS\}$ and $j \neq i$, $S[j]=S[j]+(|\Delta|*\alpha/(NS-1))$. Initially, the selection probability of each strategy is set to $1/NS$.

5 Experimental Setup

The BAP benchmark instances that have been introduced in [1] are used to validate the performance of the proposed algorithm. The benchmark involves 30 different instances (denoted as *i*1 to *i*30); each instance has 30 vessels and 13 berths [1]. In all instances, the vessel characteristics such as length and the arrival time as well as berth lengths are different. Table 2 shows the parameter settings of proposed algorithm. These settings were determined based on preliminary experiments. In this paper, we executed the proposed algorithm 31 times for each instance using different random seeds.

Table 2. The Parameter Settings

#	*Parameter*	*Value*
1	No. Of generations	500
2	Population size, *NP*	20
3	Scaling Factor, *F*	0.1
4	Crossover Rate, *CR*	0.4

6 The Computational Results

We have carried out two types of experiments. The goal of the first one is to evaluate the impact of the proposed multi-mutation operators on the performance of DE in solving BAP through a comparison with a standard, baseline DE (*DE*/*rand*/1/*bin*). The goal of second experiment is to compare the results of the proposed algorithm against the state of the art algorithms.

6.1 The Computational Comparisons of DE with and without the Game Theory Concept

In this section, we compare the computational results of DE with and without the game theory concept (denoted as DEGT and DE, respectively) using the same parameter settings, stopping condition and computer resources. Both algorithms (DEGT and DE) are executed for 31 independent runs and the results are compared using the Wilcoxon statistical test with a significance level of 0.05. The *p*-value of DEGT against DE for all instances is presented in Table 3. In this table, "+" indicates DEGT is statistically better than DE (p-value < 0.05), "-" indicates DE is statistically better than DEGT (p-value > 0.05), and "=" indicates both DEGT and DE have the same performance (p-value = 0.05). As Table 3 reflects, DEGT is statistically better than DE on 21 instances and preforms the same as DE on 2 out of 30 tested instances. This table also reveals that on 7 out of 30 tested instances, DEGT is not statistically better than DE. Although the results show that DEGT is not statistically better than DE on all tested instances, the overall finding justifies the benefit of integrating the game theory concept with DE algorithm. Indeed, the use of the game theory concept can effectively enhance the performance of DE to obtain very good results for all tested instances.

Table 3. The p-value of DEGT compared to DE

DEGT vs.	DE
Instance	*p*-value
i01	-
i02	-
i03	-
i04	-
i05	=
i06	=
i07	-
i08	-
i09	+
i10	+
i11	+
i12	-
i13	+
i14	+
i15	+
i16	+
i17	+
i18	+
i19	+
i20	+
i21	+
i22	+
i23	+
i24	+
i25	+
i26	+
i27	+
i28	+
i29	+
i30	+

6.2 The Computational Comparisons of DEGT with State of the Art Algorithms

In this section, we compare the computational results of DEGT with the current state of the art algorithms. The algorithms that we compare against are:

- Generalized set partition programming (GSPP) [18].
- Tabu search (TS) algorithm [1].
- Column generation (CG) algorithm [5].
- Clustering search (CS) [3].
- Particle swarm optimization (PSO) [4].

Table 4 gives the results of DEGT over 31 runs as well as the compared algorithms. For each instance, we present the best obtained results (best objective value) and the computational time (seconds) obtained by DEGT and the compared algorithms. In Table 4, the third column (Opt.) indicates the optimal value for each instance [18], the last row represents the average overall instances (Avg.) and boldfont indicates the best obtained results.

Table 4. The results of DEGT compared to the state of the art methods

Inst.	DEGT		GSPP		TS	CG		CS		PSO	
	Best	Time	Opt.	Time	Best	Best	Time	Best	Time	Best	Time
i01	**1409**	**10.2**	**1409**	17.92	1415	**1409**	74.61	**1409**	12.47	**1409**	11.11
i02	**1261**	**6.4**	**1261**	15.77	1263	**1261**	60.75	**1261**	12.59	**1261**	7.89
i03	**1129**	**7.1**	**1129**	13.54	1139	**1129**	135.45	**1129**	12.64	**1129**	7.48
i04	**1302**	**6.01**	**1302**	14.48	1303	**1302**	110.17	**1302**	12.59	**1302**	6.03
i05	**1207**	**4.2**	**1207**	17.21	1208	**1207**	124.7	**1207**	12.68	**1207**	5.84
i06	**1261**	**7.4**	**1261**	13.85	1262	**1261**	78.34	**1261**	12.56	**1261**	7.67
i07	**1279**	**6.5**	**1279**	14.6	**1279**	**1279**	114.2	**1279**	12.63	**1279**	7.5
i08	**1299**	**8.9**	**1299**	14.21	**1299**	**1299**	57.06	**1299**	12.57	**1299**	9.94
i09	**1444**	**3.8**	**1444**	16.51	**1444**	**1444**	96.47	**1444**	12.58	**1444**	4.25
i10	**1213**	**4.4**	**1213**	14.16	**1213**	**1213**	99.41	**1213**	12.61	**1213**	5.2
i11	**1368**	**7.2**	**1368**	14.13	1378	1369	99.34	**1368**	12.58	**1368**	10.52
i12	**1325**	**10.3**	**1325**	15.6	**1325**	**1325**	80.69	**1325**	12.56	**1325**	12.92
i13	**1360**	**10.7**	**1360**	13.87	**1360**	**1360**	89.94	**1360**	12.61	**1360**	11.97
i14	**1233**	**6.1**	**1233**	15.6	**1233**	**1233**	73.95	**1233**	12.67	**1233**	7.11
i15	**1295**	**5.7**	**1295**	13.52	**1295**	**1295**	74.19	**1295**	13.8	**1295**	8.3
i16	**1364**	**6.8**	**1364**	13.68	1375	1365	170.36	**1364**	14.46	**1364**	8.48
i17	**1283**	**4.6**	**1283**	13.37	**1283**	**1283**	46.58	**1283**	13.73	**1283**	5.66
i18	**1345**	**6.2**	**1345**	13.51	1346	**1345**	84.02	**1345**	12.72	**1345**	8.02
i19	**1367**	**9.6**	**1367**	14.59	1370	**1367**	123.19	**1367**	13.39	**1367**	11.42
i20	**1328**	**10.4**	**1328**	16.64	**1328**	**1328**	82.3	**1328**	12.82	**1328**	12.28
i21	**1341**	**6.5**	**1341**	13.37	1346	**1341**	108.08	**1341**	12.68	**1341**	7.11
i22	**1326**	**5.7**	**1326**	15.24	1332	**1326**	105.38	**1326**	12.62	**1326**	7.94
i23	**1266**	**6.7**	**1266**	13.65	**1266**	**1266**	43.72	**1266**	12.62	**1266**	7.25
i24	**1260**	**4.3**	**1260**	15.58	1261	**1260**	78.91	**1260**	12.64	**1260**	5.67
i25	**1376**	**6.2**	**1376**	15.8	1379	**1376**	96.58	**1376**	12.62	**1376**	7.13
i26	**1318**	**5.8**	**1318**	15.38	1330	**1318**	101.11	**1318**	12.62	**1318**	7.44
i27	**1261**	**4.0**	**1261**	15.52	**1261**	**1261**	82.86	**1261**	12.64	**1261**	6.16
i28	**1359**	**9.8**	**1359**	16.22	1365	1360	52.91	**1359**	12.71	**1359**	11.52
i29	**1280**	**7.1**	**1280**	15.3	1282	**1280**	203.36	**1280**	12.62	**1280**	8.11
i30	**1344**	**5.4**	**1344**	16.52	1351	**1344**	71.02	**1344**	12.58	**1344**	7.13
Avg	**1306.8**	**6.80**	**1306.8**	14.98	1309.7	1306.9	93.99	**1306.8**	12.79	**1306.8**	8.17

As can be seen from Table 4, DEGT obtained the optimal values for all tested instances. In particular, DEGT best results are the same as those produced by the GSPP. With respect to individual comparisons, DEGT best results are the same as CS, PSO, TS and CG on 30, 30 18 and 24 out of 30 tested instances, respectively. DEGT obtained better results than TS on 18 and CG on 4 instances. In addition, the average result of all instances (last row in Table 4) of DEGT is better (or the same) than the compared algorithms.

As for the computational time comparisons, Table 4 shows that, on all tested instances, the computational time of DEGT is lower than GSPP, CS, PSO, TS and CG. The overall results demonstrate that DEGT is an effective and efficient algorithm for the BAP as it obtained very good results for all tested instances within a small computational time when compared to previously reported algorithms.

7 Conclusion

In this paper, we have presented a DE algorithm to solve the BAP. DE is a population based algorithm that seeks to improve the population of solutions through the use of mutation operator(s), crossover operator and selection rule. To further enhance the performance of the DE, we coupled it with a several mutation operators in order to combine strength of different operators in one framework. Game theory is used to control the selection of which mutation operator should be used at any decision point. The computational results are carried out using the existing BAP benchmark instances. The obtained results reveal that the proposed algorithm obtained very good results when compared to DE without game theory as well as the state of the art algorithms. In addition, the computational time of proposed algorithm is lower than the compared algorithms, indicating that proposed algorithm is an effective algorithm for the berth allocation benchmark instances.

References

1. Cordeau, J.-F., Laporte, G., Legato, P., Moccia, L.: Models and tabu search heuristics for the berth-allocation problem. Transportation Science 39(4), 526–538 (2005)
2. Bierwirth, C., Meisel, F.: A survey of berth allocation and quay crane scheduling problems in container terminals. European Journal of Operational Research 202(3), 615–627 (2010)
3. de Oliveira, R.M., Mauri, G.R., Nogueira Lorena, L.A.: Clustering Search for the Berth Allocation Problem. Expert Systems with Applications 39(5), 5499–5505 (2012)
4. Ting, C.-J., Wu, K.-C., Chou, H.: Particle swarm optimization algorithm for the berth allocation problem. Expert Systems with Applications 41(4), 1543–1550 (2014)
5. Mauri, G.R., Oliveira, A.C.M., Lorena, L.A.N.: A hybrid column generation approach for the berth allocation problem. In: van Hemert, J., Cotta, C. (eds.) EvoCOP 2008. LNCS, vol. 4972, pp. 110–122. Springer, Heidelberg (2008)
6. Mallipeddi, R., Suganthan, P.N., Pan, Q.-K., Tasgetiren, M.F.: Differential evolution algorithm with ensemble of parameters and mutation strategies. Applied Soft Computing 11(2), 1679–1696 (2011)
7. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: The 2005 IEEE Congress on Evolutionary Computation, pp. 1785–1791. IEEE (2005)
8. Brest, J., Bošković, B., Greiner, S., Žumer, V., Maučec, M.S.: Performance comparison of self-adaptive and adaptive differential evolution algorithms. Soft Computing 11(7), 617–629 (2007)
9. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 13(5), 945–958 (2009)
10. Sabar, N.R., Kendall, G., Ayob, M.: An Exponential Monte-Carlo Local Search Algorithm for the Berth Allocation Problem. In: 10th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010), York, UK, August 26-29, 2014, pp. 544–548 (2014)
11. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4), 341–359 (1997)

12. Das, S., Suganthan, P.N.: Differential evolution: A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation 15(1), 4–31 (2011)
13. Sabar, N.R., Ayob, M., Kendall, G., Rong, Q.: Grammatical Evolution Hyper-Heuristic for Combinatorial Optimization Problems. IEEE Transactions on Evolutionary Computation 17(6), 840–861 (2013), doi:10.1109/TEVC.2013.2281527
14. Sabar, N.R., Ayob, M., Kendall, G., Qu, R.: A Dynamic Multiarmed Bandit-Gene Expression Programming Hyper-Heuristic for Combinatorial Optimization Problems. IEEE Transactions on Cybernetics PP(99), 1 (2014), doi:10.1109/TCYB.2014.2323936
15. Sabar, N.R., Ayob, M., Kendall, G., Qu, R.: The Automatic Design of Hyper-heuristic Framework with Gene Expression Programming for Combinatorial Optimization problems. IEEE Transactions on Evolutionary Computation PP(99), 1 (2014), doi:10.1109/TEVC.2014.2319051
16. Sabar, N.R., Kendall, G.: Aircraft Landing Problem using Hybrid Differential Evolution and Simple Descent Algorithm. Paper presented at the 2014 IEEE Congress on Evolutionary Computation (CEC 2014), pp. 520–527 (2014)