INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 0 (2016) 1–14
DOI: 10.1111/itor.12273

Evaluating the importance of randomization in adaptive large neighborhood search

Ahmad Hemmati[a] and Lars Magnus Hvattum[b]

[a]*Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, Alfred Getz veg 3, NO-7491 Trondheim, Norway*
[b]*Faculty of Logistics, Molde University College, PO Box 2110, NO-6402 Molde, Norway*
E-mail: ahmad.hemmati@iot.ntnu.no [Hemmati]; hvattum@himolde.no [Hvattum]

Received 26 November 2015; received in revised form 2 February 2016; accepted 2 February 2016

Abstract

Randomization is common in many implementations of metaheuristics, and is typically a main ingredient while considering adaptive large neighborhood search (ALNS). This paper considers a standard implementation of ALNS for maritime pickup and delivery problems, identifies seven randomized components in that implementation, and proposes and analyzes simple nonrandomized alternatives to those components. The results reveal that the randomized alternatives perform slightly better for one of seven components, the deterministic alternatives perform better for one component, while the randomized and deterministic alternatives have similar performance for the remaining five components. When analyzing runs with different initial solutions, there seems to be a larger variance in the results obtained with only randomized components, compared to the results with only deterministic components, even when the average results are similar.

Keywords: maritime transportation; routing and scheduling; statistical analysis

1. Introduction

Randomization is a main ingredient in many metaheuristics (Gutjahr, 2010; Hoos and Stützle, 2004; Sörensen and Glover, 2013), but the motivation for including randomization is not always obvious. Bartz-Beielstein et al. (2010) suggest that randomization has advantages such as possible speedups, possible gains from rerunning, avoiding cycles in the search trajectory, and achieving comparable performance while retaining a simple method. On the other hand, Glover (2007) hypothesizes that randomization is unnecessary in the context of metaheuristics, implying that reliance on randomization can prevent the development of better, deterministic, search components.

There are few theoretical results regarding the value of randomization in search algorithms. Kozen and Ruozzi (2009) established that, asymptotically, randomized algorithms cannot be better

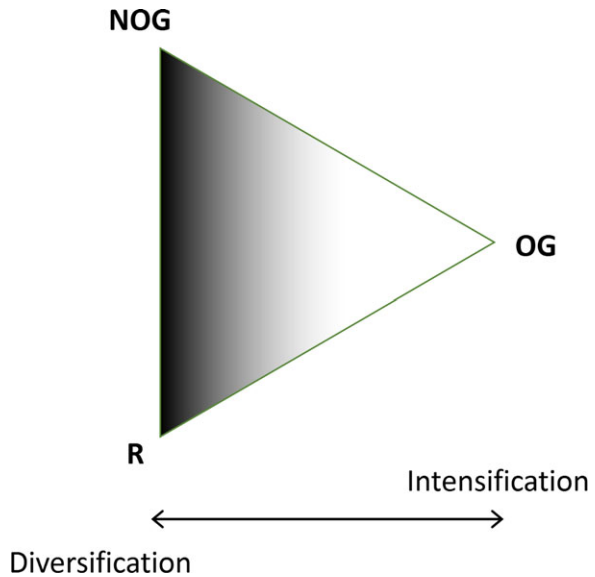

Fig. 1. A model proposed to categorize different metaheuristic components, based on whether they are random (*R*), guided by the objective function (OG), or guided by other functions (NOG), and predicting whether the component contributes to diversification (exploration) or intensification (exploitation) of the search (Blum and Roli, 2003).

than deterministic algorithms. However, while studying the presence of information constraints, Belavkin (2013) discovered situations in which optimal algorithms must be randomized. Seemingly contradictory, theoretical results could in any case be criticized as being insufficient to understand how metaheuristics work in practice, given the complex nature of the algorithms developed (Bartz-Beielstein et al., 2010; Hooker, 1994). This indicates that empirical studies should be performed to test the effect of randomization in metaheuristics.

Blum and Roli (2003) presented a model to describe the contribution of metaheuristic components in terms of diversification and intensification. The corners of the triangle in Fig. 1 correspond to components guided by randomization (*R*), the objective function (OG), or other functions (NOG). The model suggests that randomization has a role in achieving diversification, but also that randomization may possibly be replaced by auxiliary memory structures providing alternative guidance.

This paper presents a study on the role of randomization in the well-known adaptive large neighborhood search (ALNS) metaheuristic introduced by Ropke and Pisinger (2006). Empirical tests through statistical tools are used for comparing randomized search components with simple deterministic alternatives. As a basis, we use the ALNS implementation provided by Hemmati et al. (2014) for solving a set of benchmark instances of the maritime pickup and delivery problem. The implementation contains seven randomized components for which deterministic alternatives are proposed in this paper. A 2^7 factorial experiment is conducted and analyzed using both ANOVA and regression analyses. It is not our goal to discover new deterministic components that are better than the well-known and thoroughly tested randomized components, but rather to see what happens

when randomization is replaced by simple deterministic counterparts developed with minimum effort.

The remainder of this paper is structured as follows. Section 2 describes the maritime pickup and delivery problem that is solved by the ALNS heuristic. In Section 3, we provide an overview of the basic ALNS heuristic presented in Hemmati et al. (2014) and provide deterministic alternatives for all of the randomized components in the original ALNS implementation. A description of the experimental setup and the tests performed is presented in Section 4. Section 5 comprises the results obtained, and Section 6 comprises our concluding remarks.

2. Problem description

The ALNS implementations analyzed in this paper were created to obtain solutions for a maritime pickup and delivery problem. Hemmati et al. (2014) describe the problem in detail, and in the following we provide a short description followed by a mathematical formulation that formally defines the problem.

We consider a shipping company that operates a heterogeneous fleet of ships in terms of capacity, service speed, cost structure, and initial location. The problem is to determine how the fleet should service a set of known cargoes within a given planning horizon. Each cargo must be loaded at its pickup port, transported, and then unloaded at its corresponding delivery port. All loading and unloading operations must be performed within time intervals that are specific to the given cargo. Ships have given capacities and may be prevented from carrying certain cargoes due to incompatibilities. The shipping company may choose to use spot charter to transport a given cargo, instead of carrying it using one of its own ships, by paying a given lump sum. The objective is to minimize the sum of transportation costs and spot charter costs.

Let each cargo be represented by an index i and define nodes i and $i+n$—which are associated with the pickup and delivery port of cargo i—where n is the number of cargoes. The size of request i is denoted by Q_i, the set of available ships is denoted by V, and the capacity of ship $v \in V$ is K_v. Let N^P and N^D denote the set of pickup and delivery nodes and N_v the set of nodes that can be visited by ship v, which includes an origin node $o(v)$ and an artificial destination node $d(v)$. The set of arcs that ship v can traverse is denoted by A_v. Pickup and delivery nodes that can be visited by ship v are defined by the sets $N_v^P = N^P \cap N_v$ and $N_v^D = N^D \cap N_v$. A time window $[\underline{T}_i, \bar{T}_i]$ is given for each node, and sailing cost C_{ijv} and travel times T_{ijv} are given for sailing from i to j using ship v. The time at which service begins at node i using ship v is represented by the continuous variable t_{iv}, and the variable l_{iv} describes the total load onboard after completing service at node i using ship v. Binary flow variables x_{ijv} indicate whether ship v moves directly from node i to node j, while binary variables y_i indicate whether cargo i is transported by the available vessel fleet or by spot charter. If a cargo is transported using a spot charter, a cost C_i^S is incurred. The mathematical formulation, which generalizes the classic pickup and delivery problem (Ropke and Pisinger, 2006), can now be stated as follows:

$$\min \sum_{v \in V} \sum_{(i,j) \in A_v} C_{ijv} x_{ijv} + \sum_{i \in N^P} C_i^S y_i \tag{1}$$

subject to

$$\sum_{v \in V} \sum_{j \in N_v} x_{ijv} + y_i = 1, \qquad i \in N^P, \tag{2}$$

$$\sum_{j \in N_v} x_{o(v)jv} = 1, \qquad v \in V, \tag{3}$$

$$\sum_{j \in N_v} x_{ijv} - \sum_{j \in N_v} x_{jiv} = 0, \qquad v \in V, i \in N_v \backslash \{o(v), d(v)\}, \tag{4}$$

$$\sum_{j \in N_v} x_{jd(v)v} = 1, \qquad v \in V, \tag{5}$$

$$l_{iv} + Q_j - l_{jv} \leq K_v \left(1 - x_{ijv}\right), \qquad v \in V, j \in N_v^P, (i, j) \in A_v, \tag{6}$$

$$l_{iv} - Q_j - l_{(n+j)v} \leq K_v \left(1 - x_{i(j+n)v}\right), \qquad v \in V, j \in N_v^P, (i, n+j) \in A_v, \tag{7}$$

$$0 \leq l_{iv} \leq K_v, \qquad v \in V, i \in N_v^P, \tag{8}$$

$$t_{iv} + T_{ijv} - t_{jv} \leq (\bar{T}_i + T_{ijv}) \left(1 - x_{ijv}\right), \qquad v \in V, (i, j) \in A_v, \tag{9}$$

$$\sum_{j \in N_v} x_{ijv} - \sum_{j \in N_v} x_{(n+i)jv} = 0, \qquad v \in V, i \in N_v^P, \tag{10}$$

$$t_{iv} + T_{i(n+i)v} - t_{(n+i)v} \leq 0, \qquad v \in V, i \in N_v^P, \tag{11}$$

$$\underline{T}_i \leq t_{iv} \leq \bar{T}_i, \qquad v \in V, i \in N_v, \tag{12}$$

$$y_i \in \{0, 1\}, \qquad i \in N^C, \tag{13}$$

$$x_{ijv} \in \{0, 1\}, \qquad v \in V, (i, j) \in A_v. \tag{14}$$

The objective function (1) adds up the costs for operating the fleet plus the cost of spot charters. Constraints (2) ensure that all cargoes must either be picked up by a ship or transported using a spot charter. The movements of the ships are governed by constraints (3)−(5), which ensure that all ships leave the initial location, all ships that arrive at a port also leave that port, and all ships enter their final destination node, respectively. The load onboard at loading and unloading nodes is controlled through constraints (6) and (7). Constraints (8) ensure that the load does not exceed the ship capacity. Constraints (9) ensure that the time at which service starts is possible with respect to travel times. If a ship in the fleet transports a cargo, the ship must visit both loading and unloading ports, as given by constraints (10), and the loading port must be visited first, as given by constraints (11). Time windows are given by constraints (12). Finally, all flow variables and spot charter variables must take binary values, as indicated by constraints (13) and (14).

3. Description of the ALNS heuristics

The basis for the experiments performed in this work is an implementation of ALNS presented by Hemmati et al. (2014). The overall structure of the search is outlined in Algorithm 1. With a given initial solution, the search proceeds iteratively by adaptively selecting a method to remove cargoes from the current solution and a method to reinsert cargoes to repair the solution.

Algorithm 1. ALNS framework

```
Function ALNS
    generate an initial solution s with objective function of f(s)
    s_best = s, f(s_best) = f(s)
    Repeat
        s' = s
        select removal and insertion heuristics based on the search parameters
        select the number of cargoes to remove and reinsert, q
        remove q cargoes from s'
        reinsert removed cargoes into s'
        If ( f(s') ) < f(s_best) ) then
            s_best = s'
        If accept (s' , s ) then
            s = s'
        update search parameters
    Until stop-criterion met
Return s_best
```

In Hemmati et al. (2014), three different removal heuristics were used: the Shaw removal heuristic, random removal heuristic, and worst removal heuristic. The two insertion heuristics available for the ALNS were best insertion and a regret-k heuristic with k selected randomly between two and four. They terminated the algorithm after 25,000 iterations. Considering this implementation, in total seven components have been identified, which use randomization. These are summarized in Table 1.

In the following, we describe in more detail both original randomized components as well as simple, alternative deterministic components. The original implementation will, in the following, be referred to as the randomized ALNS.

Table 1
Description of the seven randomized components

Component	Description
C1	The number of cargoes that are removed in each iteration, q
C2	The selection of which removal and insertion heuristics to use in each iteration
C3	The cargoes removed by the Shaw heuristic
C4	The cargoes removed by the random removal heuristic
C5	The cargoes removed by the worst removal heuristic
C6	The value of k in the k-regret insertion heuristic
C7	The treatment of worsening solutions in the move acceptance criterion

3.1. The number of cargoes to remove in each iteration

In the randomized ALNS, the number of cargoes removed in each iteration is chosen randomly from a uniform distribution, such that the number of removed cargoes is in the interval $[q^{min}, q^{max}]$, where q^{min} is set to 4 and $q^{max} = \min(100, \xi n)$ where n is the number of cargoes and ξ is a parameter. Thus, the number of removed requests varies during the search to provide different neighborhood sizes.

To form a deterministic alternative to this, we consider that the overall goal of the selection is to remove different amounts of cargoes in each iteration. Instead of drawing the number of cargoes to remove at random, a vector q^{vec} of size $|q^{vec}| = q^{max} - q^{min} + 1$ is generated by Algorithm 2 and then component $(g \, mod \, |q^{vec}|) + 1$ of q^{vec} is considered as the number of cargoes to remove in the gth iteration. For example, if $q^{min} = 2$ and $q^{max} = 8$ then $q^{vec} = \{5,\ 4,\ 6,\ 3,\ 7,\ 2,\ 8\}$. The loop in Algorithm 2, from Lines 4–12, generates two values for q^{vec} in each iteration.

Algorithm 2. Generate q^{vec}

1: **Input** q^{min}, q^{max}
2: $q^{vec}[1] = \left\lfloor \frac{q^{min}+q^{max}}{2} \right\rfloor$
3: $a = 0,\ b = 2$
4: **Repeat**
5: $a = a + 1$
6: **If** $(q^{vec}[1] - a \geq q^{min})$ **then**
7: $q^{vec}[b] = q^{vec}[1] - a$
8: $b = b + 1$
9: **End-if**
10: $q^{vec}[b] = q^{vec}[1] + a$
11: $b = b + 1$
12: **Until** $(a < q^{max} - q^{v}[1])$
13: **Return** q^{vec}

3.2. The selection of which removal and insertion heuristics to use in each iteration

Let H be the set of heuristics from which to choose, and consider that each heuristic h is associated to a weight w_h based on its previous performance in the search. In the randomized ALNS a heuristic is selected at random, with probabilities

$$P_h = \frac{w_h}{\sum_{m \in H} w_m}. \tag{15}$$

Instead of drawing a heuristic at random, we may consider that the overall goal of the selection is to introduce a bias such that the number of times each heuristic is used depends on the past history of the search. Let r_h be the number of times that heuristic h has been used up to iteration g. In iteration g, we will then select the heuristic h that minimizes the sum of the squared differences

between the observed ratios of selection and desired ratios of selection. That is, we select a heuristic h in iteration g by

$$\arg\min_{h\in H}\left\{\left(\frac{r_h+1}{g}-P_h\right)^2+\sum_{m\in H\setminus\{h\}}\left(\frac{r_m}{g}-P_m\right)^2\right\}, \tag{16}$$

with P_h as defined in (15). In our implementation, removal heuristics and insertion heuristics are selected independently, and observed ratios of selection are reset when weights are adjusted between different segments of the search.

3.3. The cargoes selected in the Shaw heuristic

The Shaw removal heuristic is based on removing a set of similar cargoes by choosing cargoes iteratively using a relatedness measure. The relatedness is defined with respect to a distance measure, time measure, size measure, and a measure that considers the ships that can be used to serve the cargoes. The initial cargo to remove is selected at random. Additional cargoes are then selected based on the relatedness measure, but with some randomization such that more related cargoes are more likely to be selected for removal.

Based on the component classification model of Blum and Roli (2003), the main purpose of randomization is to ensure diversification. In our deterministic variant of the Shaw removal heuristic, we follow the same principle. After numbering all cargoes from 1 to n, the initial cargo to remove is selected as number $(g \ mod \ n) +1$, where g is the current iteration number and n is the number of cargoes. Additional cargoes are again selected based on the relatedness criterion. After ranking the cargoes based on relatedness, our simple deterministic alternative picks the cargo with rank $\left(g \ mod \left\lceil \frac{|L|}{2} \right\rceil\right) + 1$, where L is an array containing all requests that have not yet been removed. Thus, each time we select a cargo to be removed, the number of cargoes in L is reduced by one.

3.4. The cargoes removed by the random removal heuristic

The random removal heuristic simply selects a number of cargoes to remove at random. The purpose seems to be purely diversification, but it may not be easy to capture the same behavior by a simple deterministic rule. To attempt this, we first sort the cargoes based on the number of times that they have been removed in previous iterations. Then we select the cargo with rank $\left(g \ mod \left\lceil \rho |L| \right\rceil\right) + 1$ until the given number of cargoes to remove in each iteration has been selected. Here, based on preliminary tuning of the parameters, (ρ) is set to 0.8.

3.5. The cargoes removed by the worst removal heuristic

The worst removal heuristic removes m cargoes iteratively, making a list of the cargoes sorted according to the cost reduction induced by ignoring them, and then choosing a cargo from the list using randomization. The randomization is the same as in the Shaw heuristic, meaning that higher

ranked cargoes have a higher probability of being selected for removal. The alternative deterministic removal heuristic works similar to the alternative Shaw removal heuristic.

3.6. The value of k in the k-regret insertion heuristic

In the randomized ALNS version, the k is drawn at random between $k^{min} = 2$ and $k^{max} = 4$. A nonrandomized version of this component is simply to select $k = k^{min} + g \ mod \ (k^{max} - k^{min} + 1)$, where g is the current iteration counter.

3.7. The treatment of worsening solutions in the acceptance criterion

The initial ALNS implementation uses the move acceptance criterion from simulated annealing. That is, there is a probability of accepting a worsening move that depends on a temperature parameter and the difference in objective function value between the incumbent and new solution found. Several alternative acceptance criteria have been proposed in the literature, some of which are deterministic. We have examined the record-to-record travel criterion, originally suggested by Dueck (1993). The idea here is to accept all solutions as long as the objective function values are smaller than $z^B + D$, where z^B is the value of the best solution found so far (the record), and $D = 0.2 \left(\frac{G-g}{G}\right) z^B$ is the allowed deviation where G is the total number of ALNS iterations and g is the current iteration number. The record-to-record acceptance criterion was also used by Lei et al. (2011), with $D = 0.01 z^B$, although it was not compared to the standard simulated annealing acceptance criterion.

4. Experimental setup

In the previous section, we identified seven components of a standard ALNS implementation that relied, to at least some extent, on randomization. For each of the seven components, an alternative component was designed, aiming to replace the randomization with simple deterministic rules. To evaluate the effect of the different components, a subset of the benchmark instances presented in Hemmati et al. (2014) are used. Focusing on medium-sized instances, 50 instances for 10 different problem settings with between 22 and 60 cargoes and between 6 and 13 vessels were selected. Instances consist of both short sea and deep sea shipping instances, but only with mixed cargo sizes, thus avoiding the full load cargoes that lead to simpler routing decisions. Among five instances for each of the 10 problem settings, the first instance was used for tuning the parameters of the new nonrandom search components yielding final parameter values as specified in the previous section, whereas the rest are used in the main tests described next (in total 40 instances).

To analyze the two alternatives for each of the seven selected components, all the 40 instances will first be solved once using each of the $2^7 = 128$ different combinations of components. To determine whether the components influence the results of the search, we first perform analysis of variance (ANOVA) of type III, using as a dependent variable the accuracy of the results, measured as the improvement of the objective function of the best solution found during the search relative to a common initial solution. Three different ANOVA models are considered, one considering only the

seven components, another in which instance-specific random effects variables are added, and an extension of the latter in which we also consider interaction effects. An extension of the analysis follows in the form of a linear regression model that considers the same dependent variable, with independent variables for each component and for the instances.

After the analysis of the components, six different combinations of components are selected and used in additional computational tests. In these tests, each combination is used to solve the 40 instances 10 times, using different initial solutions. These tests may reveal different patterns regarding the performance while using different selections of components, in particular in terms of variance and mean. All statistical tests in this paper are performed with 95% confidence level.

5. Results and analysis

The first test performed was to solve each of the 40 instances using each of the $2^7 = 128$ different combinations of components. In the first ANOVA model, the response variable is equal to the relative improvement of the objective function value during the search, and the independent variables represent the effect of each of the seven components. The model indicates that the components are not important in explaining the variation in the response variable, providing an R^2 of only 0.0236. Table 2 summarizes the results of our second ANOVA model, in which the independent variables from the first model are extended by instance-specific random effects variables. This model gives an R^2 of 0.987, meaning that it explains the observed values well. However, almost the entire explanation lies in the random effects of the instances, and the effect of the different components used is far from being statistically significant. The third ANOVA model extended the second model by including interaction terms. This increased the R^2 to 0.989, but did not reveal any combination of search components that contributed significantly to the explanation of the values of the response variable.

Table 2
ANOVA with main components based on the seven alternative components and instances included as random effects

Source	Sum squares	df	Mean squares	F	Prob > F	Type
C1	0.00002	1	0.00002	0.00008	0.993	Fixed
C2	0.00001	1	0.00001	0.00013	0.991	Fixed
C3	0.00014	1	0.00014	0.00088	0.976	Fixed
C4	0.00013	1	0.00013	0.00147	0.970	Fixed
C5	0.00060	1	0.00060	0.00731	0.932	Fixed
C6	0.00003	1	0.00003	0.00042	0.984	Fixed
C7	0.00521	1	0.00521	0.04313	0.837	Fixed
Instances	25.68344	39	0.65855	9914.24423	0.000	Random
Error	0.33697	5073	0.00007			Random
Total	26.02655	5119				

The columns represent the source of the variability, sum of squares due to each source, degrees of freedom associated with each source, mean squares for each source, F-statistic, p-values, and type of source. The response variable is well explained, with an R^2 of 0.987.

Table 3
Results of a multiple linear regression model

Term	Estimate	SE	*t*-Stat	*p*-Value
Intercept	0.652	0.0008	835.138	0.000
C1	0.000	0.0002	0.518	0.604
C2	0.000	0.0002	−0.423	0.672
C3	0.000	0.0002	1.453	0.146
C4	0.000	0.0002	1.402	0.161
C5	0.001	0.0002	3.008	0.003
C6	0.000	0.0002	0.628	0.530
C7	−0.002	0.0002	−8.859	0.000

The columns represent the terms, coefficient estimates for the corresponding term, standard error of the coefficients, *t*-statistic to test whether the coefficient is statistically different from 0, and corresponding *p*-value. Not shown are the 39 instance-specific independent variables, which all have coefficients different from 0 given any reasonable significance level.

As an alternative to the ANOVA, we also ran a multiple linear regression model, with the dependent variable being the relative improvement of the objective function value during the search, and the independent variables being as in the second ANOVA model. This regression model is a standard first-order multiple linear regression model that describes a plane in the space of the dependent variables and 46 independent variables in which 39 of them are instance-specific independent variables. Total number of observations is 5120 (40×128) and the model gives R^2 of 0.986 with root mean squared error of 0.00815. Table 3 partly shows the results from the regression analysis. In this case, there are two components for which the effect is significantly different from zero. First, it is estimated that the randomized worst removal heuristic is better than the deterministic version of worst removal, with a *p*-value of 0.003. Second, the deterministic acceptance criterion is significantly better than the randomized acceptance criterion, with a *p*-value of 0. Also, the effect of the acceptance criterion is larger than the effect of the type of worst removal. For the other components, the regression analysis is consistent with the ANOVA, showing no significant effects.

The analysis above indicates that the deterministic components and randomized components perform approximately equally well. The only statistically significant exceptions are the worst removal heuristic and acceptance criterion, for which, respectively, the randomized and deterministic variants are better. When implementing a metaheuristic, it would therefore be a suggestion to use the deterministic acceptance criterion, randomized worst removal, and select any of the two variants for the other components.

To further investigate which combination of components is actually better, we perform additional tests. Consider first Fig. 2, which ranks the 128 different combinations of components used in the tests described so far. Deterministic components are indicated by colored areas, whereas randomized components are indicated by white areas. In these tests, each combination of components was used to solve each of 40 instances once. Sorting these results based on the average of objective value divided by the initial solution, where the initial solutions are the same for all the sets, illustrates that there are few trends regarding whether deterministic or randomized components are better. However, it is clear that the deterministic acceptance criterion (component number 7) is more frequently occurring to the left (being better) than the randomized acceptance criterion.

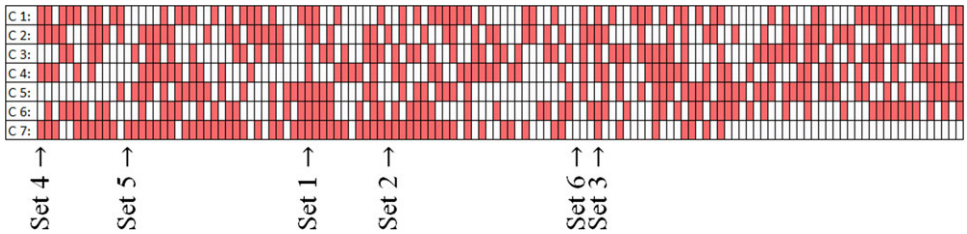

Fig. 2. The place of six selected sets among all sorted 128 different sets (combinations) of components based on the improvement in objective value (top ranked set is found to the left).

Table 4
Description of the six selected sets of components

Sets	Description
Set 1	All components are deterministic
Set 2	Based on the signs of coefficient in regression regardless of significance (C2 and C7 are deterministic)
Set 3	All components are deterministic except C5 based on significant and sign in regression
Set 4	Best set of components in our first sample (C1, C2, C4, and C7 are deterministic and C3, C5, and C6 are random)
Set 5	All components are random except C7
Set 6	All components are random

Table 5
Performance of the six selected sets in 10 runs of 40 instances with five different sizes

		Set 1		Set 2		Set 3		Set 4		Set 5		Set 6	
No. of cargoes	No. of ships	Min. gap[a]	Avg. gap[a]	Min. gap[a]	Avg. gap[a]	Min. gap[a]	Avg. gap[a]	Min. gap[a]	Avg. gap[a]	Min. gap[a]	Avg. gap[a]	Min. gap[a]	Avg. gap[a]
22	6	0.04	0.32	0.00	0.29	0.00	0.19	0.00	0.15	0.00	0.27	0.00	1.04
23	13	0.25	0.40	0.00	0.17	0.00	0.18	0.00	0.14	0.01	0.12	0.00	0.23
30	6	0.05	0.74	−0.20	0.27	−0.11	0.50	−0.23	0.38	−0.08	0.48	−0.15	0.99
35	7	0.39	0.90	0.05	0.90	0.15	0.98	0.07	0.79	−0.07	0.60	0.16	1.21
60	13	1.36	2.49	0.44	1.80	0.75	2.34	0.39	2.09	0.34	1.84	−0.05	1.52
Average		0.42	0.97	0.06	0.69	0.16	0.84	0.05	0.71	0.04	0.66	−0.01	1.00
Variance		0.65		0.54		0.92		0.76		0.55		1.79	

[a]Average of minimum and average gap to best known (%) in Hemmati et al. (2014) for both short sea and deep sea.

To give more depth to the analysis, six different sets of components were selected, as described in Table 4, for further testing. Using these six sets of components, the same 40 instances were solved again, but using 10 different starting solutions. Results are summarized in Table 5. It is now clear that the purely deterministic implementation (Set 1) is worse than the partly or fully randomized implementations when considering the best solution found across the 10 starting solutions. However, on average, they are quite similar, as was found in the previous results.

Furthermore, among 400 runs (10 runs of 40 instances) of each set, both best and worst performances belong to Set 6 (all components randomized). The best performance is −0.7 % and the worst performance is +13.9%, where the performance is calculated based on the gap to the best-known

Table 6
Overview of some statistics and running time of the six selected sets

	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
Number of zero or negative gaps (out of 40 × 10 runs)	123	150	156	156	165	128
Number of negative gaps (out of 40 × 10 runs)	4	31	14	21	16	19
Number of contributions to new best known (out of 14)	2	4	2	4	6	6
Average time (seconds)	41	45	46	47	41	39

Table 7
New best-known solutions to benchmark instances and the contribution of each set. The black colored cells indicate which sets contributed to find the new best known solutions.

Sr. no.	Instance	Old best known	New best known	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
1	SHORTSEA_MUN_C30_V6_HE_2	4,568,421	4,549,708		■		■	■	■
2	SHORTSEA_MUN_C30_V6_HE_3	4,106,293	4,098,111		■	■	■	■	■
3	SHORTSEA_MUN_C35_V7_HE_2	4,543,650	4,533,265	■					
4	SHORTSEA_MUN_C60_V13_HE_2	8,055,970	8,036,365						■
5	SHORTSEA_MUN_C60_V13_HE_3	7,651,685	7,619,449		■				
6	DEEPSEA_MUN_C30_V6_HE_2	16,896,623	16,785,044				■		
7	DEEPSEA_MUN_C30_V6_HE_3	21,298,546	21,183,928				■		
8	DEEPSEA_MUN_C35_V7_HE_2	54,839,181	54,830,688					■	
9	DEEPSEA_MUN_C35_V7_HE_3	56,258,180	56,182,502	■	■	■		■	
10	DEEPSEA_MUN_C35_V7_HE_4	61,489,997	61,354,812					■	
11	DEEPSEA_MUN_C35_V7_HE_5	63,943,961	63,904,705			■			■
12	DEEPSEA_MUN_C60_V13_HE_2	75,878,996	75,656,934						■
13	DEEPSEA_MUN_C60_V13_HE_4	91,076,203	90,884,252					■	
14	DEEPSEA_MUN_C60_V13_HE_5	89,721,702	89,092,714						■

solutions of Hemmati et al. (2014). Based on the paired *t*-tests, with 95% confidence level, on each pair of the selected six sets, we can divide the sets into two groups based on their performance. The first group, including Sets 2, 4, and 5, performs better than the second group, which includes Sets 1, 3, and 6. Inside the two groups, we found no statistically significant differences.

The results of the tests with the six sets of components also led to finding some new best results for the benchmark instances used. Table 6 shows the number of new best-known solutions found by using each of the sets, as well as the number of times that the previous best-known results were equaled or bettered. Table 7 gives detailed values for the new best-known solutions as well as the contribution of each set to find the new best-known solution.

Among all six sets, Set 5 performs best in almost all criteria. It has the greatest number of zero or negative gaps in 400 runs compared to other sets and in terms of the number of new best-known solutions it places in the first rank together with Set 6. The minimum average of average gaps and minimum variance also belong to this set and the minimum average of minimum gaps for this set is very close to the best set. Besides Set 5, which uses the deterministic acceptance criterion, Sets 1−4, which have also deterministic components instead of original randomized components, have less variance. Among these sets, Sets 2 and 4 are significantly better than Set 6, in which all components are random.

While the randomized components have been used in many studies, and have had the opportunity to be refined over time, our results indicate that initial implementations of straightforward deterministic alternatives are immediately able to match the performance of the randomized components. This may encourage researchers to perform additional studies of the effect of the randomized search components, and possibly to find deterministic components that can surpass their performance, once their detailed function has been understood.

6. Concluding remarks

Randomization is featured in most current-day implementations of metaheuristics. Little work has been done to understand the role of randomization from a theoretical perspective, and empirical results aiming to analyze randomization are hard to find in the metaheuristics literature. This paper aimed to perform an analysis of the ALNS metaheuristic, where randomization is a common ingredient.

Seven components of a recent ALNS implementation were identified as relying on randomization. For each component, a simple nonrandom alternative was implemented. A full factorial design was implemented to analyze the two different alternatives of each component. Slightly different results were obtained while using ANOVA and a linear regression. While ANOVA indicated that there were no statistically significant differences between the random and nonrandom alternative for any of the components, the regression indicated that the nonrandom move acceptance criterion outperformed the standard simulated annealing move acceptance criterion. On the other hand, the randomized version of the worst removal heuristic seemed slightly better than the nonrandomized alternative.

Finally, six combinations of components were selected for additional tests, with 10 runs per instance, to give further insight into the performance of the ALNS alternatives. These runs resulted in 14 new best-known results on the benchmark problem instances used. It also revealed that the combination with only randomized components had a different performance than the combination with only deterministic components, even though their average performance was very similar. The use of only randomized components led to larger variance in the results providing, across the 10 runs for each instance, both larger maximum gaps and smaller minimum gaps compared to the deterministic components. Together with the results from the four other combinations, it seems that this effect was mainly due to the move acceptance criterion, and the other components are of lesser importance. This may give an indication that a better understanding of the move acceptance criterion in ALNS may be the key to improving the performance of this metaheuristic, while the degree of randomization for the other components may be of lesser importance.

Acknowledgments

This research was carried out with financial support from the Research Council of Norway through the DOMinant II project. The authors wish to thank anonymous reviewers whose comments helped to improve the paper.

References

Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M., 2010. Introduction. In Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (eds) *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, Berlin, Heidelberg, pp. 1–16.

Belavkin, R.V., 2013. Optimal measures and Markov transition kernels. *Journal of Global Optimization* 55, 387–416.

Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys* 35, 268–308.

Dueck, G., 1993. New optimization heuristics—the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics* 104, 86–92.

Glover, F., 2007. Tabu search—uncharted domains. *Annals of Operations Research* 149, 89–98.

Gutjahr, W.J., 2010. Stochastic search in metaheuristics. In Gendreau, M., Potvin, J.-Y. (eds) *Handbook of Metaheuristics* (2nd edn), Vol. 146, *International Series in Operations Research & Management Science*. Springer, New York, pp. 573–597.

Hemmati, A., Hvattum, L.M., Norstad, I., Fagerholt, K., 2014. Benchmark suite for industrial and tramp ship routing and scheduling problems. *INFOR* 52, 28–38.

Hooker, J.N., 1994. Needed: an empirical science of algorithms. *Operations Research* 42, 201–2012.

Hoos, H.H., Stützle, T., 2004. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann Publishers, San Francisco, CA.

Kozen, D., Ruozzi, N., 2009. Applications of metric coinduction. *Logical Methods in Computer Science* 5, 1–19.

Lei, H., Laporte, G., Guo, B., 2011. The capacitated vehicle routing problem with stochastic demands and time windows. *Computers and Operations Research* 38, 1775–1783.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science* 40, 4, 455–472.

Sörensen, K., Glover, F., 2013. Metaheuristics. In Gass, S., Fu, M. (eds) *Encyclopedia of Operations Research and Management Science* (3rd edn). Springer, London, pp. 960–970.

Sörensen, K., Schittekat, P., 2013. Statistical analysis of distance-based path relinking for the capacitated vehicle routing problem. *Computers and Operations Research* 40, 3197–3205.