RK-EDA: A Novel Random Key Based Estimation of Distribution Algorithm

Mayowa Ayodele(✉), John McCall, and Olivier Regnier-Coudert

Robert Gordon University, Aberdeen, UK
{m.m.ayodele,j.mccall,o.regnier-coudert}@rgu.ac.uk

Abstract. The challenges of solving problems naturally represented as permutations by Estimation of Distribution Algorithms (EDAs) have been a recent focus of interest in the evolutionary computation community. One of the most common alternative representations for permutation based problems is the Random Key (RK), which enables the use of continuous approaches for this problem domain. However, the use of RK in EDAs have not produced competitive results to date and more recent research on permutation based EDAs have focused on creating superior algorithms with specially adapted representations. In this paper, we present RK-EDA; a novel RK based EDA that uses a cooling scheme to balance the exploration and exploitation of a search space by controlling the variance in its probabilistic model. Unlike the general performance of RK based EDAs, RK-EDA is actually competitive with the best EDAs on common permutation test problems: Flow Shop Scheduling, Linear Ordering, Quadratic Assignment, and Travelling Salesman Problems.

Keywords: Estimation of distribution algorithm · Random key · Permutation problems · Cooling scheme · Univariate model

1 Introduction

Estimation of Distribution Algorithms (EDAs) are Evolutionary Algorithms (EAs) that generate solutions by sampling a Probabilistic Model (PM) of promising solutions. The ability to model the features of more promising solutions is a major attribute that differentiates them from most other EAs [7]. They benefit from the use of machine learning techniques, which makes them better at solving certain categories of larger and more difficult problems [12]. Problems naturally represented as permutations have however been identified as challenging for EDAs. This is attributed to the fact that EDAs have not been extensively explored to solve this class of problems [3]. EDAs for permutation spaces have therefore been a focus of research in recent years.

EDAs applied to permutations have been categorised into ad hoc approaches with varying strategies, integer space based and continuous space based [3]. One of the common continuous representations for solving permutations in EAs is the well-known Random Key (RK). RKs have an advantage over most other permutation representations as they always produce permutation feasible solutions.

This is particularly not the case for integer based EDAs as they often require a procedure to handle the mutual exclusivity constraint.

RK based EDAs have however been considered the poorest [3] of the EDAs designed for permutation problems. RK representation has not been sufficiently adapted to benefit from the operation of EDA. It contains some inherent redundancy as a result of several RKs producing the same ordering thereby introducing plateaux to the search space [2,3,13]. Also, variability in the values that capture the same priority across solutions of a population limits the information captured by the probabilistic model. They therefore struggle to produce competitive results [7]. Models that are more specific to permutations such as histogram models [16,17], permutation distribution models [4–6] and factoradics [14] have shown better performances.

Some classical examples of RK based EDAs are REDA [15], EGNA$_{ee}$ &UMDA$_c$ [10]. REDA uses the triangulation of Bayesian network approach and focuses on model efficiency by modelling subset nodes of a problem. EGNA$_{ee}$ builds a Gaussian network where the structure of a problem is learnt using edge exclusion tests [10]. The UMDA$_c$ which is also a structure identification algorithm based on Gaussian network performs hypothesis tests to identify the density of its model's components. In addition, IDEA-ICE [2] can also be classified as a RK based EDA, although it uses a crossover operator to preserve building blocks in addition to its probabilistic model. Also, RKs associated with the building blocks are rescaled to improve the likelihood of them being properly combined. The IDEA-ICE shows better performance compared to the classical RK based EDAs.

The proposed Random Key Estimation of Distribution Algorithm (RK-EDA) attempts to capture some of the identified limitations of RKs as well as exploit their advantages.

The rest of this paper is described as follows. Section 2 motivates and describes the novel algorithm, RK-EDA. A discussion of problem sets and experimental design is presented in Sect. 3. Section 4 presents and discusses results while conclusions are presented in Sect. 5.

2 RK-EDA

The proposed RK-EDA is a univariate EDA whose probabilistic model, similar to UMDA$_c$, is based on mean values of genes in more promising solutions of a population. It exploits already found good genes by sampling a Gaussian distribution based on mean and variance values. Unlike UMDA$_c$, RK-EDA imposes a user defined variance parameter rather than a population generated one. This is because we achieved better performance using a controlled variance value. Furthermore, we propose to use a cooling rate parameter to control exploration and exploitation. This controls the level of variance in solutions of a population such that there is more exploration at the start of the algorithm, which automatically cools as the search progresses.

In this section, we present the algorithmic details of RK-EDA.

Algorithm 1. RK-EDA

```
1: Initialise $\sigma$, $t_s$ and $p_s$
2: Generate initial population $P$ of size $p_s$
3: for $g = 1$ to $MaxGen$ do
4:    Evaluate and rescale individuals in $P$
5:    Select best $t_s < p_s$ solutions to form $S$
6:    Calculate $\mu_S$
7:    $c = 1 - \frac{g}{MaxGen}$
8:    $\sigma_g = \sigma * c$
9:    $M = N(\mu_S, \sigma_g)$
10:   $P_{new} = \emptyset$
11:   repeat
12:      Sample $M$ to generate offspring $off$
13:      Add $off$ to $P_{new}$
14:   until $|P_{new}| = p_s$
15:   $P = P_{new}$
16: end for
```

As shown in Algorithm 1, RK-EDA requires the initialisation of three parameters which are initial variance σ, truncation size t_s and population size p_s. Since the stopping criteria is based on the number of fitness evaluations allowed (*FEs*), the maximum number of generations $MaxGen$ is estimated by dividing *FEs* by p_s.

A population P of RKs is randomly generated, evaluated and rescaled. The rescaling procedure requires the conversion of RKs to ranks e.g. [0.12, 0.57, 0.23, 0.25, 0.99] becomes [1, 4, 2, 3, 5]. The ranks are then rescaled to values between 0 and 1. This is done by setting $rescaledRK_i = \frac{rank_i - 1}{n-1}$ where $rescaledRK_i$ and $rank_i$ are respectively the rescaled RK and rank of gene i, and n is the problem size. The RK in the previous example therefore becomes [0.00, 0.75, 0.25, 0.50, 1.00]. With this approach, another set of RKs [0.01, 0.06, 0.03, 0.04, 0.2] which is the same solution as the previous example will have the same rescaled RK value [0.00, 0.75, 0.25, 0.50, 1.00]. With this approach, we are able to minimise redundancy and improve the information captured by the probabilistic model.

The best t_s solutions of the population are selected to generate a population S. Also, μ_S in ln. (6) is an array $\mu_{S_1}, ..., \mu_{S_n}$ that saves the mean of all RKs at indexes $\{1 \cdots n\}$ in the selected population S. Note that μ_{S_n} refers to the mean of all RKs in the n^{th} index of each solution of S.

Cooling Rate c is calculated with respect to the particular generation such that its value is higher for the first few iterations and lower at the last set of iterations. As shown in ln. (8), c is used to generate generational variance σ_g. Multiplying c with σ to form σ_g makes it possible to achieve higher exploration at the start of the algorithm and more exploitation as g increases.

Furthermore, M is defined as a normal distribution $N(\mu_S, \sigma_g)$ and is updated for each generation g. Unlike μ_S which is an array of values, σ_g is not an array but a single value. An offspring solution *off* is generated by sampling M. Each gene i ($1 \leq i \leq n$) of *off* is generated based on σ_g and μ_{S_i}, *off* is repeatedly added to the offspring population P_{new} until its size equals p_s. At the end of each generation, P_{new} completely replaces the parent population P.

3 Experimental Settings

In this section we present the permutation problem instances as well as the parameter settings.

3.1 Permutation Problems

To assess the performance of RK-EDA, we apply it to a range of permutation benchmark problems. These problems include Flow Shop Scheduling Problem (FSSP), Linear Ordering Problem (LOP), Quadratic Assignment Problem (QAP) and Travelling Salesman Problem (TSP). These are formerly defined in [3], we have used the same objective functions as presented in the review paper and is summarised in Table 1. Note that we also consider the more recently used Total Flow Time (TFT) criteria for further experiments on the FSSP.

Table 1. Definition of the permutation problems

PPs	Objective functions	Definition of symbols
TSP	$min\left\{\sum_{i=2}^{n} d_{c_{i-1},c_i} + d_{c_n,c_1}\right\}$	c_i - i_{th} city d_{c_{i-1},c_i} - distance between c_{i-1} and c_i
FSSP	$min\{c_{j_n,m}\}\ c_{j_i,m} = max(c_{j_i,m-1}, c_{j_{i-1},m}) + p_{j_i,m}$	j_i - i_{th} job m - machine m $c_{j_i,m}$ - completion time for j_i on m $p_{j_i,m}$ - processing time for j_i on m
QAP	$min\left\{\sum_{i=1}^{n}\sum_{j=1}^{n} h_{a,b} \times d_{l_a,l_b}\right\}$	l_i - i_{th} location $h_{a,b}$ - flow between facilities a and b d_{l_a,l_b} - distance between l_a and l_b
LOP	$max\left\{\sum_{i=1}^{n}\sum_{j=1}^{n} d_{\omega_i\omega_j}\right\}$	ω_i - index of row and column at position i Matrix $D = [d_{ij}]$

3.2 Problem Sets

We evaluate RK-EDA using the selected permutation problems in [14]. We acknowledge that many of the problems are small instances especially the FSSP. Also, results from running RK-EDA on the FSSP problem instances gives an intuition that the algorithm is more competitive on the FSSP. We therefore added four larger FSSP problems.

The problem sets used in this paper are listed below.

1. TSP: *bays29, berlin52, dantzig42* and *fri26*
2. FSSP: *tai20-5-0, tai20-5-1, tai20-10-0* and *tai20-10-1* (smaller instances) *tai50-10-0, tai50-10-1, tai100-20-0* and *tai100-20-1* (larger instances)
3. QAP: *tai15a, tai15b, tai40a* and *tai40b*
4. LOP: *t65b11, be75np* and *be75oi*

These are commonly used problems and we consider them useful for comparing with other EDAs for permutation problems.

3.3 Parameter Setting

To be able to understand the parameter settings that suit RK-EDA, we explored a range of values and found different parameters suitable for different problem classes and sizes. To be able to make a fair comparison between RK-EDA and the considered algorithms, we use a set of parameters across all problems as done in the review [3]. The set of parameters used for RK-EDA is shown in Table 2. Based on preliminary tests, these parameters produce relatively good quality solutions across all problem classes and instances.

Table 2. Parameter values for RK-EDA

Parameter	Values
Population Size (p_s)	50
Truncation Size (t_s)	0.1*p_s
Variance (σ)	$1/(3.14 * log_{10}n)$
Stopping Criteria	$1000n^2$ FEs
Maximum Number of Generations ($MaxGen$)	$20n^2$
Number of Runs	10

4 Results and Discussion

In this section, we present the results of running RK-EDA on the aforementioned permutation problem sets. Table 3 shows the minimum, maximum, average and standard deviation based on 10 runs of RK-EDA using the parameters presented in Table 2. Results are compared based on averages and standard deviations. We have highlighted results where optimal solution was found (appended *). We also highlight results that are significantly better (appended ✓) or not significantly different (appended **) from the best of the reviewed algorithms. We used the student t-test to measure statistical significance with a 0.05 significance level.

The results in Table 3 are presented according to problem classes. Note that FSSP$_s$ and FSSP$_l$ respectively denote the smaller and larger instances of the FSSP.

Table 3. Average performance of RK-EDA on benchmark problems

Groups	Problems	Minimum	Maximum	Mean	Stdev
TSP	**bays29**	**2020.0**	**2091.0**	**2041.5**	**21.3***
	berlin52	8207.0	8742.0	8404.6	164.0
	dantzig42	729.0	824.0	771.2	35.6
	fri26	**937.0**	**968.0**	**949.5**	**11.9***
FSSP$_s$	**tai20-5-0**	**1278.0**	**1279.0**	**1278.1**	**0.3* ✓**
	tai20-5-1	**1359.0**	**1360.0**	**1359.5**	**0.5****
	tai20-10-0	1586.0	1618.0	1602.9	11.1
	tai20-10-1	1680.0	1691.0	1685.2	3.2
FSSP$_l$	**tai50-10-0**	**3046.0**	**3119.0**	**3090.7**	**24.2****
	tai50-10-1	**2923.0**	**2964.0**	**2937.6**	**14.9 ✓**
	tai100-20-0	**6344.0**	**6424.0**	**6386.4**	**21.0 ✓**
	tai100-10-1	**6291.0**	**6381.0**	**6338.6**	**27.2 ✓**
QAP	tai15a	393496.0	412072.0	404616.6	5350.2
	tai15b	51968294.0	52238818.0	52088443.6	72876.7
	tai40a	3353650.0	3418792.0	3391139.0	20951.9
	tai40b	642257062.0	659424886.0	652079961.9	4690584.3
LOP	t65b11	355180.0	356311.0	356028.2	295.6
	be75np	**716221.0**	**716930.0**	**716644.3**	**249.8****
	be75oi	110928.0	111156.0	111012.3	77.8

Table 4 shows the performance of each algorithm on the considered problems. The table is ordered according to the overall ranks shown in column "ALL". Columns TSP, FSSP$_s$, QAP, LOP and FSSP$_l$ show the average ranks of algorithms on instances of their respective problem classes. Column ALL is the average rank of algorithms on all instances of TSP, FSSP$_s$, QAP and LOP. Since one of the motivations for selecting the additional problems (FSSP$_l$) is that we ranked relatively high on FSSP$_s$, FSSP$_l$ was not used to create the overall rank so as to eliminate bias towards performance on FSSP. Also, since one of the reviewed algorithms was not applied to instances of FSSP$_l$, it will be impossible to generate an overall rank for the algorithm. To generate the ranks shown in the table, we use the average fitness recorded by each algorithm as reported in [3] and [14] as well as that of RK-EDA shown in Table 3. All algorithms are ranked from best to worst for each problem.

We used "-" to denote missing results where authors have not applied their algorithm to the given problem class.

According to the review presented in [3], EHBSA$_{WT}$ and NHBSA$_{WT}$ were recognised as the best performing algorithms. A similar result is depicted by the overall rank of these algorithms in Table 4. EHBSA$_{WT}$ ranks 1st while RK-EDA ranks 2nd with NHBSA$_{WT}$.

Table 4. Average ranks of algorithms

Algorithms	TSP	FSSP$_s$	QAP	LOP	ALL	FSSP$_l$
EHBSA$_{WT}$ [16]	1.00	1.75	4.00	2.00	2.13	3.25
RK-EDA	3.75	2.50	7.00	2.25	4.00	1.00
NHBSA$_{WT}$ [17]	8.50	3.00	2.00	1.75	4.00	3.00
NHBSA$_{WO}$ [17]	6.00	4.50	2.50	4.25	4.27	4.75
Factoradics [14]	6.50	6.25	6.75	7.00	6.47	-
UMDA [9]	8.25	6.75	4.75	6.25	6.53	7.00
EBNA$_{BIC}$ [1]	8.25	7.50	3.75	6.50	6.67	7.00
EHBSA$_{WO}$ [16]	2.25	6.00	10.00	10.75	7.27	9.75
MIMIC [1]	10.50	8.00	6.25	6.50	7.80	3.50
TREE [13]	12.25	10.50	8.75	9.75	10.33	7.00
IDEA-ICE [2]	11.25	10.75	10.50	9.75	10.53	8.75
REDA$_{UMDA}$ [15]	14.50	11.00	12.00	11.50	12.27	12.25
REDA$_{MIMIC}$ [15]	8.50	14.25	14.00	13.25	12.47	12.25
EGNA$_{ee}$ [11]	9.00	14.75	13.25	15.00	12.93	12.25
omeGA [8]	14.25	12.00	14.75	14.50	13.80	14.75
UMDA$_c$ [11]	10.25	16.00	15.75	15.00	14.13	13.50

We observed that the RK based EDAs such as REDA$_{UMDA}$, REDA$_{MIMIC}$, EGNA$_{ee}$, UMDA$_c$ as well as the RK based GA (OmeGA) are ranked least in Table 4 which is similar to the conclusion in the review Ceberio et al. [3]. OmeGA had been introduced in the review to compare with the performance of the EDAs in general. RK-EDA however shows a different trait outperforming all other RK based algorithms.

Furthermore, the performance of RK-EDA varies with different classes of problems. It produced competitive results on the FSSP, ranking 2^{nd} on FSSP$_s$ and 1^{st} on FSSP$_l$. RK-EDA produced statistically better results than the best of the reviewed algorithms on three FSSP$_l$ instances. It also produced competitive results for the TSP and LOP but much less competitive performance on the QAP. This may be attributed to the fact that parameters that suit other problem classes are not particularly suitable for the search space presented by the QAP.

In addition to the reviewed algorithms, other permutation based EDAs exist but were not included in the previous comparison because their results are not reported on the selected problems. GM-EDA [4] exhibits the best results on FSSP when hybridised with local search procedures such as variable neighbourhood search (VNS). We therefore compare RK-EDA with GM-EDA on a selected set of FSSP instances. In order to compare the two EDAs in a fair way, we use the reported results of GM-EDA without VNS.

We use the same set of parameters presented by the authors in [4] except that we do not consider elitism. This is because preliminary experiments show that

Table 5. Parameter values and stopping criteria for experiments on FSSP based on TFT

Parameter settings:	Parameter	Values
	Population size (p_s)	$10n$
	Truncation size (t_s)	$0.1{*}p_s$
	Variance (σ)	0.15
	$MaxGen$	FEs/p_s
	Number of runs	20
Stopping criteria:	Problem sizes	FEs
	20×05	182224100
	20×10	224784800
	50×10	256208100
	100×20	283040000

Table 6. Average TFT for FSSP

Problems	Algorithm	Average	Stdev
tai20-5-0	RK-EDA	14085	14
	GM-EDA	**14058**	**13**
tai20-5-1	RK-EDA	15223	20
	GM-EDA	15224	46
tai20-10-0	RK-EDA	21003	14
	GM-EDA	21006	46
tai20-10-1	RK-EDA	22660	81
	GM-EDA	**22561**	**135**
tai50-10-0	RK-EDA	89233	292
	GM-EDA	89041	400
tai50-10-1	RK-EDA	84858	138
	GM-EDA	84849	326
tai100-20-0	**RK-EDA**	**373607**	**523**
	GM-EDA	374708	1388
tai100-20-1	**RK-EDA**	**379947**	**501**
	GM-EDA	380750	868

elitism does not improve the performance of RK-EDA. In addition, 0.15 initial variance value particularly produced competitive results for FSSP instances. Table 5 shows the parameters of RK-EDA, which are adapted for solving the FSSP.

In Table 6, we present the average fitness over 20 runs for RK-EDA as well as GM-EDA. The results are based on the Total Flow Time (TFT) objective function and we compare using instances of FSSP$_s$ and FSSP$_l$. The results for GM-EDA have been extracted from [4]. Values that are significantly better are presented in bold. The results show that the GM-EDA is significantly better on two of the smallest problems (*tai20-5-0* and *tai20-10-1*) while RK-EDA shows significant improvement on the largest problems (*tai100-20-0* and *tai100-20-1*). There are however no significant difference between the performance of the algorithms on other instances.

Results from comparing RK-EDA with GM-EDA as shown in Table 6 also indicate that RK-EDA is competitive and should be further explored to solve bigger and more complex problems.

5 Conclusions

EDAs based on RKs have previously been considered the poorest of permutation based EDAs [3]. One of the problems posed by RKs is attributed to the variety of ways of representing an ordering [13]. In this paper, we introduce a novel RK based EDA (RK-EDA) which addresses this by rescaling the RKs uniformly. This approach improves the information captured by the probabilistic model. Furthermore, RK-EDA uses a cooling scheme to manage the rate of exploration/exploitation of the search space such that there is better exploration at the start of the algorithm and better exploitation of already found good pattern as the search progresses.

Furthermore, learning a probability structure is considered the most expensive operation in EDAs [2], we present a simple model, which only saves the mean of solutions in a selected population. This is relatively computationally efficient. RK-EDA whose procedure is comparatively simple produces very competitive results. It outperforms other reviewed continuous EDAs. It is also competitive with the best permutation EDAs in general.

RK-EDA's most competitive performance is seen on FSSP and the least on QAP. It's performance on FSSP gets more competitive as the problem size increases presenting the best results on the largest of the considered FSSP instances. The performance of RK-EDA on larger problems is therefore recommended for further investigation.

In addition, the use of local search has been reported to improve the performance of the GM-EDA, hybridisation of the RK-EDA may also improve its performance.

References

1. Bengoetxea, E., Larrañaga, P., Bloch, I., Perchant, A., Boeres, C.: Inexact graph matching by means of estimation of distribution algorithms. Pattern Recogn. **35**(12), 2867–2880 (2002)
2. Bosman, P.A., Thierens, D.: Crossing the road to efficient IDEAs for permutation problems. In: Proceedings of the 6th annual conference on Genetic and evolutionary computation, pp. 219–226. ACM (2001)

3. Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J.A.: A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. Prog. Artif. Intell. **1**(1), 103–117 (2012)
4. Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J.A.: A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. IEEE Trans. Evolut. Comput. **18**(2), 286–300 (2014)
5. Ceberio, J., Mendiburu, A., Lozano, J.A.: The plackett-luce ranking model on permutation-based optimization problems. In: 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 494–501. IEEE (2013)
6. Ceberio, J., Mendiburu, A., Lozano, J.A.: Kernels of mallows models for solving permutation-based problems. In: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, pp. 505–512. ACM (2015)
7. Hauschild, M., Pelikan, M.: An introduction and survey of estimation of distribution algorithms. Swarm Evolut. Comput. **1**(3), 111–128 (2011)
8. Knjazew, D., Goldberg, D.E.: Omega-ordering messy GA: solving permutation problems with the fast messy genetic algorithm and random keys. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 181–188 (2000)
9. Larrañaga, P., Etxeberria, R., Lozano, J.A., Peña, J.M.: Optimization in continuous domains by learning and simulation of Gaussian networks. In: Workshop in Optimization by Building and using Probabilistic Models, pp. 201–204. A Workshop withing the 2000 Genetic and Evolutionary Computation Conference, GECCO 2000, Las Vegas, Nevada, USA (2000)
10. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, vol. 2. Springer, New York (2002)
11. Lozano, J., Mendiburu, A.: Estimation of distribution algorithms applied to the job shop scheduling problem: Some preliminary research. In: Larrañaga, P., Lozano, J.A. (eds.) Estimation of Distribution Algorithms, pp. 231–242. Springer, Heidelberg (2002)
12. Pelikan, M., Sastry, K., Cantú-Paz, E.: Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications. Springer, Heidelberg (2006)
13. Pelikan, M., Tsutsui, S., Kalapala, R.: Dependency trees, permutations, and quadratic assignment problem. In: Genetic and Evolutionary Computation Conference: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, vol. 7, pp. 629–629 (2007)
14. Regnier-Coudert, O., McCall, J.: Factoradic representation for permutation optimisation. In: Bartz-Beielstein, T., Branke, J., Filipič, B., Smith, J. (eds.) PPSN 2014. LNCS, vol. 8672, pp. 332–341. Springer, Heidelberg (2014)
15. Romero, T., Larrañaga, P.: Triangulation of Bayesian networks with recursive estimation of distribution algorithms. Int. J. Approx. Reason. **50**(3), 472–484 (2009)
16. Tsutsui, S.: Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 224–233. Springer, Heidelberg (2002)
17. Tsutsui, S., Pelikan, M., Goldberg, D.E.: Node histogram vs. edge histogram: a comparison of PMBGAs in permutation domains. MEDAL Report (2006009) (2006)