AUTHOR(S): ANKRAH, R., REGNIER-COUDERT, O., MCCALL, J., CONWAY, A. and HARDWICK, A

TITLE: Performance analysis of GA and PBIL variants for real-world location-allocation problems.

YEAR: 2018

Publisher citation: ANKRAH, R., REGNIER-COUDERT, O., MCCALL, J., CONWAY, A. and HARDWICK, A. 2018. Performance analysis of GA and PBIL variants for real-world location-allocation problems. In Proceedings of Institute of Electrical and Electronics Engineers (IEEE) congress on evolutionary computation (IEEE CEC 2018), 8-13 July 2018, Rio de Janeiro, Brazil. New York: IEEE [online], article ID 8477727. Available from: https://doi.org/10.1109/CEC.2018.8477727

OpenAIR citation: ANKRAH, R., REGNIER-COUDERT, O., MCCALL, J., CONWAY, A. and HARDWICK, A. 2018. Performance analysis of GA and PBIL variants for real-world location-allocation problems. TIn Proceedings of Institute of Electrical and Electronics Engineers (IEEE) congress on evolutionary computation (IEEE CEC 2018), 8-13 July 2018, Rio de Janeiro, Brazil. New York: IEEE, article ID 8477727. Held on OpenAIR [online]. Available from: https://openair.rgu.ac.uk

Performance Analysis of GA and PBIL Variants for Real-World Location-Allocation Problems

Reginald Ankrah
School of Computing and Digital Media
Robert Gordon University
Aberdeen, Scotland
r.b.ankrah@rgu.ac.uk

Olivier Regnier-Coudert
School of Computing and Digital Media
Robert Gordon University
Aberdeen, Scotland
o.regnier-coudert@rgu.ac.uk

John McCall
School of Computing and Digital Media
Robert Gordon University
Aberdeen, Scotland
j.mccall@rgu.ac.uk

Anthony Conway
Research and Innovation Department
Adastral Park, British Telecommunications Plc
Ipswich, England
anthony.conway@bt.com

Andrew Hardwick
Research and Innovation Department
Adastral Park, British Telecommunications Plc
Ipswich, England
andrew.hardwick@bt.com

***Abstract*—The Uncapacitated Location-Allocation problem (ULAP) is a major optimisation problem concerning the determination of the optimal location of facilities and the allocation of demand to them. In this paper, we present two novel problem variants of Non-Linear ULAP motivated by a real-world problem from the telecommunication industry: Uncapacitated Location-Allocation Resilience problem (ULARP) and Uncapacitated Location-Allocation Resilience problem with Restrictions (ULARPR). Problem sizes ranging from 16 to 100 facilities by 50 to 10000 demand points are considered. To solve the problems, we explore the components and configurations of four Genetic Algorithms [1], [2], [3] and [4] selected from the ULAP literature. We aim to understand the contribution each choice makes to the GA performance and so hope to design an Optimal GA configuration for the novel problems. We also conduct comparative experiments with Population-Based Incremental Learning (PBIL) Algorithm on ULAP. We show the effectiveness of PBIL and GA with parameter set: random and heuristic initialisation, tournament and fined_grained tournament selection, uniform crossover and bitflip mutation in solving the proposed problems.**

Key words: uncapacitated facility location problem; uncapacitated facility location resilience problem; uncapacitated facility location resilience problem with restrictions; genetic algorithm; population-based incremental learning algorithm.

I. INTRODUCTION

Location-Allocation Problems (LAP) are concerned with allocating locations to a set of facilities to service a set of customers in such a way as to optimise a cost function, subject to a set of constraints. LAP has been well-researched, and there are many formulations. LAP is, in general, combinatorial problems and so the number of solutions increases exponentially with problem size, defined by the number of facilities and customers. It is well-known that LAP are NP-hard [1] and so approximate optimisation algorithms are typically applied for large problem instances. Many metaheuristic methods have been proposed for LAP, including tabu search (TS) [5], simulated annealing (SA) [6], hybridization of simulated annealing and random descent method [7], Variable neighborhood search.[8], Genetic algorithms (GA) [9], [10][11],[12],[13],[1],[2],[3],[4], [14], [15],[16], Bees algorithm [14], Clustering search (CS) method using Simulated annealing (SA) [17], hybrid Particle swarm optimisation algorithm [18], hybrid intelligent algorithms [19].

One important LAP constraint is whether or not each facility has a limited capacity for the number of customers it can serve. We can, therefore, classify LAP into capacitated problems (CLAP), where the capacity constraint applies, and uncapacitated problems (ULAP) where the limitation does not apply, that is each facility is unconstrained in the number of customers that it can service. In this paper, we are principally interested in ULAP, and in particular, in two novel non-linear ULAPs motivated by a real-world problem from the telecommunications industry. The new problems are described in Section II.

Among the most successful algorithms applied to ULAP are genetic algorithms, in particular the work presented in [1], [2], [3] and [4]. Although all four approaches have the basic core of a GA, they differ in component and configuration choices. These are presented in section III-B. Our aim in this paper is to understand the contribution each choice makes to the GA performance and so hope to design an optimal GA configuration for the novel problems.

GAs are known to be good at locating global optimum due to their ability to explore a function space from multiple points in parallel [20]. However, their convergence rate usually is slower than conventional optimisation techniques. Moreover, the problem of genetic drift (one of the mechanisms by which the population converges to a single member due to the stochastic nature of the selection operator [21]) can cause loss of diversity within the population. Once diversity is lost, the crossover operator becomes inefficient in traversing the search

space. Although mutation can be used to add diversity in the population, its impact is limited [21].

For this reason, we also explore the effectiveness of Population-Based Incremental Learning (PBIL) [22] on ULAP. PBIL is a simple Estimation of Distribution Algorithm (EDA). EDAs iteratively construct and sample probabilistic models of good solutions and have been efficiently applied to challenging combinatorial problems in recent years. Our motivation in selecting PBIL is twofold: it has the potential benefits of an EDA but with a lightweight (univariate) modelling cost; and we observe that specifically for the novel ULAP formulations proposed, useful problem knowledge can be encoded directly into the probabilistic model. To the best of our knowledge, this is the first time PBIL has been applied to ULAP.

The paper is organised as follows: ULAP problem definitions are presented in section II. A detailed description of the selected algorithms is presented in section III. Section IV describes our experiments and an analysis of the results. Our final conclusions are presented in section V.

II. Problem definitions and Mathematical models

A. Uncapacitated Location-Allocation problem (ULAP)

Let $A = a_1, a_2, ...a_m$ be a set of *m* facility sites, and $B = b_1, b_2, ...b_n$ be a set of n customers. The aim of ULAP is to connect all customers in B to selected sites from A so that every customer is connected to exactly one facility. The service cost between customer b_j and location a_i is defined as d_{ij} and a facility opening cost c_i is assigned to every facility location a_i. In ULAP, facilities are assumed to provide an unlimited amount of service, that is they can serve any number of customers.

The objective function shown as equation (1) minimizes the sum of overall service costs and fixed costs of establishing a facility. Constraint (2) ensures that a customer can only be assigned to one facility while constraint (3) reflects the binary nature of variables x_{ij} where $x_{ij} = 1$ means customer b_j is connected to facility a_i located at potential site a_i. $x_{ij} = 0$ means otherwise.

$$\textbf{Min} \sum_{i=1}^{m}(c_i * min(1, \sum_{j=1}^{n} x_{ij})) \quad + \quad \sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij} d_{ij} \quad (1)$$

Subject to:

$$\sum_{i=1}^{m} x_{ij} = 1, \forall j \quad (2)$$

$$x_{ij} \in \{0, 1\}, \forall i, j \quad (3)$$

B. Two real-world variants

In this paper, we propose two additional variants to ULARP, inspired by real-world cases. We consider a service company that supplies data services to its customers. The company needs to decide optimal locations out of a set of m facilities sites to establish new facilities in addition to existing ones to effectively supply customers demand within a specified region. In addition to the cost c_i for opening a new facility, a running facility cost of h_i is considered. Also, the company needs to decide which existing facilities to shut down due to unprofitability and where to reassign customers serviced by a facility if the decision is made to close it down. There is an associated cost of e_i for shutting down a facility and a reassignment cost of g_{ik} for moving customer b_j from facility a_i to facility a_k.

In these real-world variants, the overall bandwidth demand at a specific facility impacts the associated cost. To accommodate all the customers allocated to a facility, its core bandwidth needs to be adapted, resulting in a cost calculated by the function $f_i(x)$, which is a step cost function. The core bandwidth to be used at a facility is the smallest value from the set $S = \{s_1, s_2, ..., s_q\}$ of all core bandwidths available that can accommodate the sum of all facility's customer expected bandwidths. Each core bandwidth is associated with a cost p_q.The expected bandwidth of a customer b_j is denoted as w_j

Finally, to ensure resiliency and uninterrupted service supply, the company needs to decide on the best backup connection for customers. The objective of the company is to minimise the total costs of opening new facilities, shutting down profitless facilities, reassigning customers, service cost of main customer connections, service cost of customers backup connections and the running costs of facilities.

Two non-linear problem variants are presented. The first model is not restricted to the number of facilities that can be opened to service customer demand. In the second problem, an additional constraint on the maximum number of opened facilities is considered.

1) Uncapacitated Location-Allocation Resilience Problem (ULARP): ULARP aims to connect all customers in B to selected sites from A so that every customer is connected to one main facility and a second facility that serves as a backup connection.

As expressed in Equation (4), the objective is to minimise the total cost of opening facilities, cost of closing facilities, cost of moving customers between facilities, service cost subject to main customer allocation, service cost subject to backup customer allocation, facility running cost and bandwidth-related costs.

Additional decision and problem variables that were not considered in ULAP are introduced.The cost of moving a customer b_j from facility a_i to facility a_k is denoted as g_{ik}, while the service cost of customer b_j to backup facility a_i is denoted as d_{ij}.

$$\begin{aligned}\textbf{Min} & \sum_{i=1}^{m}((c_i + h_i) * min(1, \sum_{j=1}^{n} x_{ij})) \\ + & \sum_{i=1}^{m}(e_i * max(0, 1 - \sum_{j=1}^{n} x_{ij})) \\ + & \sum_{j=1}^{n}\sum_{i=1}^{m}\sum_{k=1}^{m} \frac{(g_{ik} * |x_{ij} - b_{jk}|)}{2} \\ + & \sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij}d_{ij} + \sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij}d_{ij} + \sum_{i=1}^{m} f_i(x_{ij})\end{aligned} \tag{4}$$

where:

$$f_i(x_{ij}) = \begin{cases} p_1 & \sum_j^n w_j x_{ij} < s_1 \\ p_2 & s_1 \leq \sum_j^n w_j x_{ij} < s_2 \\ \dots & \\ p_q & s_{q-1} \leq \sum_j^n w_j x_{ij} \end{cases} \tag{5}$$

Subject to:

$$\sum_{i=1}^{m} x_{ij} = 2, \forall j \tag{6}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \tag{7}$$

2) Uncapacitated Location-Allocation Resilience Problem with Restrictions (ULARPR): Unlike ULARP, which has no restrictions on the number of facilities that can be opened ULARPR presents restriction on facilities to be opened. To maintain operating convenience and economic reasons of budget constraint the company decides to limit the maximum number of facilities that can be opened to Δ facilities among the set of all facility sites. The objective remains the same as the objective of ULARP presented in (4). However, we introduce a new constraint shown in (8) that limits the maximum number of facilities. In this paper and based on the real-world case study, Δ is set to 34.

$$\sum_{i=1}^{m} min(1, \sum_{j=1}^{n} x_{ij}) \leq \Delta \tag{8}$$

III. Proposed Methods

In this section, we describe the algorithms used in the paper and the choices made to configure and parameterise their various components. All of the methods use a common problem representation and so can be directly compared to the same search space.

A. Problem Representation

Possible solutions to the ULAP are assessed based on the facility and service costs. Nevertheless, considering the capacity of the facilities are not limited, once the locations have been determined, the highest performance is achieved when each customer employs the facility giving the least expensive service. Therefore for ULAP, a solution to the problem is wholly determined by the choices to place a facility or not at each of the m locations.

We therefore select a binary representation, denoted $Q = (q_1, q_2, ..., q_m)$ of length *m* where m is the number of locations. Any subset *A'* of the facility locations corresponds to a unique solution $Q(A')$ defined as follows:

$$Q(A')_i = \begin{cases} 1; & \textit{when } a_i \in A' \\ 0; & \textit{otherwise} \end{cases} \tag{9}$$

That is, $Q(A')_i = 1$ when a facility is placed at potential site a_i, and $Q(A')_i = 0$ otherwise.

For a given solution $Q(A')$, we may calculate the optimal service cost d_j^* for customer b_j as

$$d_j^* = min\{d_{i,j} | a_i \in A'\} \tag{10}$$

Customer b_j is therefore allocated to facility j with service cost d_j^*.

B. Genetic Algorithm

In this subsection, we describe the different choices for configuring components and parameters of a Genetic Algorithm (GA) that we have drawn from the ULAP literature. For reference, pseudocode showing how the components combine as a GA is presented as Algorithm 1.

Initialisation: Two choices are available for initialisation of the population of solutions: *Random* initialisation, which samples representation space with uniform probability and *Heuristic* initialisation, which uses the *ClassInitialiaser* method described in [1]. Heuristic initialisation uses a classification index *t* which is the ratio of facility costs to service costs to generate an initial population focused on regions of the search range where optimal solutions are anticipated to be.

Selection: Three selection methods are employed ; Tournament Selection [1]; Fine-Grained Tournament Selection (FGTS) [3][4]; and Roulette-Wheel Selection [2]. FGTS allows tournaments of different sizes to be used for selection within a single evolutionary cycle. This is controlled by a real-valued parameter F_{tour}, which is the desired average tournament size.

CrossOver: Two crossover operators are used: One Point Crossover [3][2][4] and Uniform Crossover [1].

Mutation: Two mutation operators are used: Bitflip Mutation [3][2][4] [1]; and Partial Space Search [1]. Partial Space Search searches the solution space around the mutating solution that is likely to contain a good solution.

Algorithm 1 Genetic Algorithm pseudo-code

(1) initialise population of solutions;
(2) set termination criteria to **false**;
while (termination criteria are **false**) **do**
  (3) select solutions for reproduction;
  (4) apply crossover;
  (5) apply mutation;
  (6) evaluate the fitness of new solutions;
  (7) update population;
  (8) evaluate termination criteria;
end while
(9) return population;

C. Population-Based Incremental Learning

Population-based Incremental Learning (PBIL) was originally introduced by Baluja [22] [23]. PBIL is a method that merges simple GAs with competitive learning. The crossover operator is taken away in PBIL, and the population role is redefined. PBIL keeps a real-valued probability vector (*pv*) from which solutions are generated. The *pv* controls the arbitrary bitstrings generated by PBIL, and it's used to create other individuals through learning. Learning in PBIL consists of using the current probability distribution to create *N* individuals [24]. The generated individuals are assessed based on an objective function. The *pv* is then updated using the fittest individuals. This increases the likelihood of generating individuals that are alike to the fittest individuals. The *pv* of PBIL is often initialised with a 0.5 probability, ensuring an equal chance of producing a 1 or 0. During the exploration process, the values in the *pv* divert from 0.5, towards 0.0 or 1.0. The PBIL implementation used for this experiment was presented by Baluja in [25]. The pseudo-code of PBIL is presented below.

Algorithm 2 Population-based Incremental learning Algorithm pseudo-code

1 Initialise probability vector pv with 0.5;
while (!StopCondition()) **do**
  2 Generate a population using the pv;
  3 Evaluate solutions and find best and worst individuals based on fitness;
  4 Update *pv* towards "Best" individual.
  5 Update *pv* away from "Worst" individual;
  5 Perform mutation on *pv*;
  6 Generate a new population using updated pv;
end while

IV. Numerical Experiments

Both GAs and PBIL parameters were set following preliminary experiments. The values thus determined for the GAs are crossover rate of 0.9, a mutation rate of 0.2. Parameters for PBIL includes positive learning rate of 0.2, the negative learning rate of 0.054, mutation probability of 0.02 and mutation amount of 0.05. Both GAs and PBIL have a population size of 50. Though the four main GAs presented in literature used different fitness evaluations, we employ the standard fitness evaluation budget for all GAs. Different fitness evaluations are set for the different problems based on preliminary experiments. 20000 fitness evaluations for ULAP and 5000 for ULARP and ULARPR. In the results presented we present each possible parameter permutation of a GA under *parameter set (ps)*. Each *ps* is represented as number/initialiser/selection/crossover/mutation where number represent one of the 24 parameter permutation of a GA, initialiser can be random (r) or ClassInitialiser (c), selection can be tournament (t), fined-grained tournament (f) or roulette wheel (rw), crossover can be uniform (u) or one-point (1p) and mutation can be bitflip (b) or partial space search (p).

The base problem (ULAP) presented in section II-A was solved using the benchmark problem set from OR-Library [26]. The problem set constitutes benchmark instances for comparing computational efficiencies of different algorithms for ULAP. It is made up of 15 instances ranging from small to large instances. Small instances include Cap71 to Cap74 which has 16 facilities by 50 demand or customer points. The medium instances include Cap101 to Cap104 which contains 25 facilities and 50 demand points and Cap131 to Cap134 containing 50 facilities and 50 demand points. The largest of these instances are CapA to CapC which is made up of 100 facilities by 1000 demand points. Below we present and discuss the ranking results for the best five *ps* on ULAP. We then compare the performance of the best *ps* against PBIL on ULAP.

.

Table I presents the ranking results for the best 5 *ps* for all 15 instances of ULAP. The results presented for each problem instance is average over 20 runs. The ranking is obtained using the Friedman ranking test. Under data, we present all 15 instances of ULAP. Mean gives the overall average over 20 runs of each *ps* on all 15 problem instances. "std" is the standard deviation obtained. Mean rank gives the overall ranking average for all *ps* on all 15 problem instances whiles "pos" indicates the overall performance of a *ps*. If all results obtained for a problem instance are highlighted in bold, this means that the optimal value for the problem instance was obtained for all 20 runs by each algorithm.

From table I, it can be seen that *ps* 2/c/t/u/b gives the best results 10 out of 15 times with a mean ranking of 4.63 based on the Friedman ranking test with a 99% confidence level. However, in comparing the mean results using the Unpaired t-test with a confidence level of 95%, the differences in results obtained by the first five parameter permutations are not statistically significant, meaning that the results produced by all five algorithms on ULAP within a set budget of 20000 fitness evaluations are satisfactory. ps 2/c/t/u/b is the same parameter combination of components presented in [1] where similar good results are recorded. Out of the top 5 parameter permutations, the first 3 initialised their solutions classInitialiser. This shows that on the basic linear problem classInitialiser can estimate the number of facilities to be opened hence

TABLE I
RANKING RESULTS OF GAS

Data	2/c/t/u/b	6/c/t/u/p	10/c/f/u/b	1/r/t/u/b	9/r/f/u/b
Instance 71	**9.33E+05**	**9.33E+05**	**9.33E+05**	**9.33E+05**	**9.33E+05**
Instance 72	**9.78E+05**	**9.78E+05**	**9.78E+05**	**9.78E+05**	**9.78E+05**
Instance 73	**1.01E+06**	**1.01E+06**	**1.01E+06**	**1.01E+06**	**1.01E+06**
Instance 74	**1.03E+06**	**1.03E+06**	**1.03E+06**	**1.03E+06**	**1.03E+06**
Instance 101	7.97E+05	7.97E+05	7.97E+05	**7.97E+05**	7.97E+05
Instance 102	**8.55E+05**	8.55E+05	**8.55E+05**	**8.55E+05**	**8.55E+05**
Instance 103	8.94E+05	8.94E+05	8.94E+05	8.94E+05	**8.94E+05**
Instance 104	**9.29E+05**	**9.29E+05**	**9.29E+05**	**9.29E+05**	**9.29E+05**
Instance 131	7.95E+05	7.94E+05	7.95E+05	7.94E+05	**7.94E+05**
Instance 132	**8.52E+05**	8.52E+05	8.52E+05	8.52E+05	8.52E+05
Instance 133	**8.94E+05**	8.94E+05	8.94E+05	8.94E+05	8.94E+05
Instance 134	9.30E+05	**9.29E+05**	9.29E+05	9.29E+05	9.31E+05
Instance A	**1.72E+07**	1.74E+07	1.73E+07	1.75E+07	1.80E+07
Instance B	**1.31E+07**	1.31E+07	1.32E+07	1.32E+07	1.33E+07
Instance C	1.18E+07	1.20E+07	**1.18E+07**	1.20E+07	1.21E+07
Mean	**3.54E+06**	3.56E+06	3.54E+06	3.58E+06	3.62E+06
std	5.54E+06	5.59E+06	5.56E+06	5.63E+06	5.73E+06
Mean Rank	**4.63**	5.60	5.67	5.77	6.40
Pos	**1st**	2nd	3rd	4th	5th

focusing the search in an area where there is the likelihood of finding good solutions. All five algorithms employed uniform crossover making it the best crossover parameter for ULAP. This is because uniform crossover maintains good genes of individuals whiles exploring the search space. Tournament selection and Bitflip mutation proved to be the ideal selection and mutation operators for the basic ULAP problem. All in all, the combination of these four parameters offers the best results regarding solution accuracy and reliability on the basic ULAP.

In table II, a comparative result is presented between the best GA and PBIL (PBIL is initialised with a probability vector of 0.5). The figures in parentheses in the first column under data show the test instance. *Ps* show the parameter set of the best GA and PBIL. *Freq* Shows the number of times the algorithm achieves the known optimum value for the problem instance. Average relative percentage deviation *(ARPD)* is a special form of standard deviation expressed as a percentage which tells whether the regular standard deviation of results is small or large when compared to the know optimum mean. It is calculated as

$$ARPD = (\sum_{i=0}^{n} \frac{((Algorithm_i - Best) * 100)}{Best}) \qquad (11)$$

Where *Best* is the known optimum. *Average* shows the mean value of 20 runs for each data instance. *Sd* Shows the standard deviation from the mean over 20 runs. Optimal is the known optimum value of each problem instance. *Best* and *worst* are, respectively, the best and worst values obtained by each algorithm on each data instance out of 20 runs. The *Fes* is the total number of solutions evaluated. *Fes* is used as the stopping condition for the algorithms for each run. *SD of Fes* is the standard deviation from the mean of fitness evaluation over 20 runs. The parameters and figures highlighted in bold are the best parameter set, corresponding average and best *Fes* respectively.

Results of GA and PBIL on ULAP presented in II shows a healthy performance of the two algorithms on the presented data instances (Cap71, 102, 132 & A). The table shows that optimal solutions are achieved by both GA and PBIL with high accuracy for the relatively small instance. However, this accuracy seems to drop with larger instances. Still, GA outperforms PBIL by the number of frequency that it obtained the optimal values for medium and large instances. Even though the difference of average results achieved by both algorithms is considered to be not statistically significant based on statistical testing (Unpaired t-test), GA completed almost all it's running in less *Fes* than PBIL. Making GA the best of the two on the basic ULAP in-terms of solution accuracy and reliability. A run distribution diagram of GA and PBIL for *Cap71, Cap132 and CapA* are presented in fig. 1, 2 & 3.

Fes in fig 1 & 2 is considered not statistically significant for the small and medium problem instance. The difference, however, grows bigger with an exponential increase in the problem size. This is seen in fig. 3. This is evident that GA performs better than PBIL on larger instances for the basic ULAP problem. The poor performance of PBIL in comparison to GA on ULAP can be attributed to the random

TABLE II
COMPARATIVE RESULTS BETWEEN GA AND PBIL

Data	Ps	Freq.	Arpd (%)	Ave	Sd	Best	Worst	Fes	SD of Fes.
71	**2/c/t/u/b**	20	0.00	**9.33E+05**	0.00	9.33E+05	9.33E+05	**4.88E+02**	1.59E+02
	PBIL	20	0.00	9.33E+05	0.00	9.33E+05	9.33E+05	5.83E+02	1.72E+02
102	**2/c/t/u/b**	20	0.00	**8.55E+05**	0.00	8.55E+05	8.55E+05	**1.85E+03**	1.77E+03
	PBIL	19	0.01	8.55E+05	2.76E+02	8.55E+05	8.56E+05	2.65E+03	4.29E+03
132	**2/c/t/u/b**	16	0.01	**8.52E+05**	1.98E+02	8.51E+05	8.51E+05	**6.98E+03**	7.32E+03
	PBIL	13	0.05	8.52E+05	7.87E+02	8.51E+05	8.55E+05	1.06E+04	7.78E+03
A	**2/c/t/u/b**	11	0.88	**1.73E+07**	4.48E+05	1.72E+07	1.81E+07	**1.30E+04**	7.76E+03
	PBIL	4	2.00	1.75E+07	3.86E+05	1.72E+07	1.84E+07	1.82E+04	4.42E+03

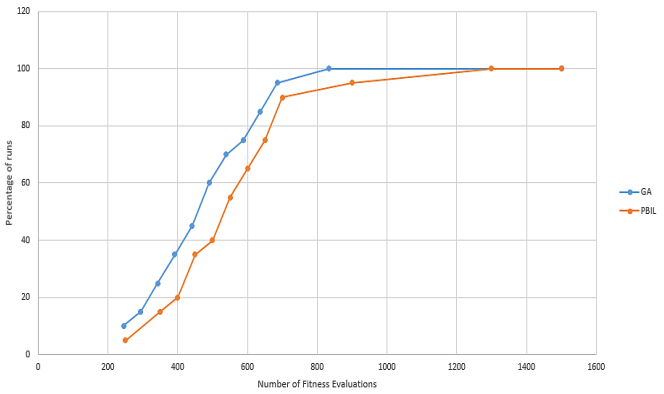

Fig. 1. Run length distribution of Cap 71

Fig. 3. Run length distribution of Cap A

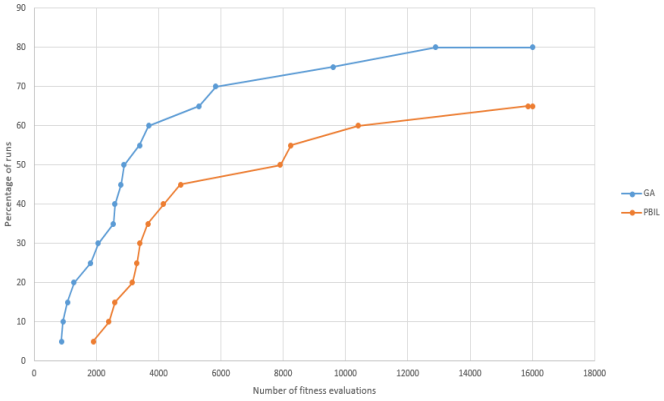

Fig. 2. Run length distribution of Cap 132

initialisation (0.5) of PBIL. Because the search is unfocused due to the random initialisation, PBIL oscillates for a while and then converges to an optimum value due to some random factor. GA, on the other hand, makes use of uniform crossover and bitflip mutation to improve on solution quality.

The problem described in sections II-B1 and II-B2 are new variants inspired by real-world cases with no known optimal values. The problem set includes a single problem instance with 100 facilities by 10000 demand points. Since results of GAs *ps* and PBIL are non-deterministic, every benchmark instance is solved 20 times by each algorithm. To be able to access the performance of each algorithm, a budget of 5000 *Fes* is used. The set budget is based on results of preliminary experiments. For table III and table IV, *Rank* indicates the overall performance of an algorithm. Each table holds three different experiments conducted using different probability values to initialise PBIL and GAs.

In the first experiment, it is assumed that there is no prior knowledge of the problem hence we initialised PBIL using a probability vector of 0.5, and the same value is used to initialise GA ps with a random initialiser. Giving each facility a 50% chance of being opened.

In the second experiment, we introduce some problem knowledge into the initialisation process. We are restricted to a maximum number of 34 facilities to open however we do not know which facilities are to be opened. Hence we initialise PBIL with a probability vector of 0.34, and the same is done for ps with a random initialiser. This is to help focus the search of the algorithms within a specified area of the search space where good solutions for the problem are likely to exist.

In the third experiment, we assume that there already exist 29 opened facilities and there is an associated high cost of shutting any of these facilities down. We are only allowed to have a maximum of 34 facilities opened. To help ensure that

the already opened facilities stay open to avoid incurring the cost of shutting down, we set a stronger probability vector for the first 29 facilities of 0.85 chance of staying opened and the remaining five facilities get a 0.15 chance of being opened. The same setting is used for ps with random initialisation.

Results obtained for the Top 5 Algorithms on ULARP and ULARPR are presented in tables III & IV respectively.

TABLE III
COMPARATIVE RESULTS ON ULARP

Ps	Freq	Ave.	Sd	Rank
ULARP with 0.5				
1/r/t/u/b	15	**9.91E+07**	1.86E+04	1st
PBIL	15	9.91E+07	3.42E+04	2nd
9/r/f/u/b	12	9.91E+07	3.41E+04	3rd
10/c/f/u/b	1	9.93E+07	2.68E+05	4th
13/r/f/u/p	1	9.94E+07	2.46E+05	5th
ULARP with .34				
9/r/f/u/b	20	**9.91E+07**	0	1st
1/r/t/u/b	18	9.91E+07	4.15E+04	2nd
PBIL	16	9.91E+07	4.21E+04	3rd
10/c/f/u/b	1	9.93E+07	2.68E+05	4th
13/r/f/u/p	1	9.94E+07	2.13E+05	5th
ULARP with .85 & 0.15				
9/r/f/u/b	19	**9.91E+07**	8.73E+03	1st
1/r/t/u/b	19	9.91E+07	2.87E+04	2nd
PBIL	12	9.91E+07	1.89E+05	3rd
10/c/f/u/b	1	9.93E+07	2.68E+05	4th
13/r/f/u/p	1	9.94E+07	2.29E+05	5th

TABLE IV
COMPARATIVE RESULTS ON ULARPR

Ps	Freq	Ave.	Sd	Rank
ULARPR with 0.5				
1/r/t/u/b	1	**1.61E+08**	1.08E+06	1st
2/c/t/u/b	1	1.61E+08	8.75E+05	2nd
9/r/f/u/b	1	1.62E+08	2.08E+06	3rd
10/c/f/u/b	1	1.63E+08	2.38E+06	4th
PBIL	1	1.64E+08	1.17E+06	5th
ULARPR with .34				
1/r/t/u/b	1	**1.61E+08**	3.16E+05	1st
9/r/f/u/b	1	1.61E+08	9.02E+05	3nd
2/c/t/u/b	1	1.61E+08	8.75E+05	3rd
13/r/f/u/p	1	1.62E+08	1.62E+06	4th
10/c/f/u/b	1	1.63E+08	2.38E+06	5th
ULARPR with .85 & 0.15				
1/r/t/u/b	1	**1.61E+08**	6.99E+05	1st
2/c/t/u/b	1	1.61E+08	8.75E+05	2nd
9/r/f/u/b	1	1.62E+08	1.49E+06	3rd
10/r/f/u/p	1	1.63E+08	2.38E+06	4th
13/r/f/u/p	1	1.63E+08	1.62E+06	5th

Results of ULARP presented in tab:table3 shows *ps*1/r/t/u/b and ps 9/r/f/u/b outperforming all other algorithms. A look at these two best algorithms shows similarities in the parameters the employ; random initialisation, uniform crossover and bitflip mutation. These parameters combine to offer the best results on ULARP. Fine-grained tournament selection is employed in three of the top five algorithms for each of the initialisations making it the best selection method for tackling ULARP. This is because Fine-grained tournament selection allows the ratio between exploration and exploitation to be set precisely. This is essential as the ratio between exploration and exploitation governs the search process in a GA. It is very effective on large problem size as it allows tournaments with a different number of competitors to be held within one step of the selection. This allows for a good chance for the best individuals in the population to be selected for mating [27]. PBIL shows good performance and provides satisfactory results for ULARP on all initialisations.

Similar results are observed in table IV, with *ps*: 1/r/t/u/b outperforming all other algorithms. However, PBIL failed to appear in the top five of 3 initialisations for ULARPR. The overall performance of PBIL on ULARP and ULARPR can be attributed to the small fixed learning rate. Due to the nature of the problem size, a small fixed learning rate allows PBIL to explore better the search space which in turn introduces diversity into the population. However, this often comes at a cost as the diversity often delays the convergence of PBIL. Varying the learning rate for PBIL as the problem size changes can be essential to ensure a good trade-off between exploitation and exploration.

TABLE V
COMPARISON OF PBIL RESULTS ON ULARP AND ULARPR

pv	Freq	Ave.	Sd
ULARP			
0.5	15	9.91E+07	3.42E+04
0.34	16	**9.91E+07**	4.21E+04
0.85	12	9.91E+07	1.89E+05
ULARPR			
0.5	1	**1.64E+08**	1.17E+06
0.34	1	1.64E+08	1.51E+06
0.85	1	1.66E+08	2.14E+06

Table V gives the performance of PBIL on ULARP and ULARPR. Results achieved for all three initialisations for ULARP is considered as not statistically significant. The best result was achieved when PBIL was initialised with a probability vector of 0.34 where it achieved an 80% frequency rate. The worst result for the three initialisations was obtained when PBIL was initialised with a probability vector of 0.85 & 0.15. This could be attributed to the biased initialisation where a stronger chance was given to the first 29 facilities to stay open. This means that if the cost of connecting customers to the first 29 facilities is costly, then that will affect the overall results obtained by PBIL on ULARP.

Results obtained by PBIL using *pv* of 0.5 and 0.34 is considered to be not statistically significant. However, results achieved when PBIL is initialised with a pv of 0.85 & 0.15 is considered to be statistically significant in comparison to pv of 0.5 and 0.34. This is because even though PBIL is initialised with a pv of 0.5 or 0.34. PBIL is allowed to find the best facilities out of the defined scope to open. When initialised with a pv of 0.85 & 0.15, the strong biased drive of keeping the first 29 facilities open affects the overall results obtained.

V. Conclusion and Future Work

In this paper, we compare the performance of 24 parameter permutation of a GA and PBIL on the basic linear ULAP problem and two new Non-linear version of ULAP with three different initialisation probability. Computational experiments on all 16 problem instances show that for the basic ULAP, *ps*: 2/c/t/u/b offers the best performance. This is because the ClassInitializer can estimate the number of facilities required to be opened and configures the initial population in the area of the search space where a good solution is likely to exist. Uniform crossover helps maintain the good genes of individuals whiles exploring the search space. It can produce acceptable solutions concerning solution quality and reliability.

For more extensive problems of ULARP and ULARPR *ps*:1/r/t/u/b and ps 9/r/f/u/b produces acceptable solutions concerning solution quality and reliability and hence offers the best performance for solving the new problem variants. On all 3 *pv* initialiasations, the results obtained by ps:1/r/t/u/b and ps 9/r/f/u/b are considered to be statistically insignificant. The satisfactory performance of *ps*:1/r/t/u/b, *ps*: 9/r/f/u/b and PBIL on a large data instance of 10000 demand points show that they can be extended to tackle more significant data problems.

In this paper, we focused on the static variants of ULAP which are formulated with consideration to some current parameter values involving facility costs and demand level. However, if changes can be defined for such values, then planning with consideration to future adjustment in locating facilities to serve changing demand becomes an important factor. For this reason, future work will focus on modelling the dynamic variant of ULAP where the evaluation of a fixed solution incorporates dynamic or time-varying aspects of the problem.

References

[1] Tohyama H, Ida K, Matsueda J. A genetic algorithm for the uncapacitated facility location problem. Electronics and Communications in Japan. 2011;94(5):47–54.

[2] Mavani K, Shah M. Synthesis of silver nanoparticles by using sodium borohydride as a reducing agent. International Journal of Engineering Research & Technology. 2013;2(3).

[3] Marić M. An efficient genetic algorithm for solving the multi-level uncapacitated facility location problem. Computing and Informatics. 2012;29(2):183–201.

[4] Korac VM, Kratica J, Savić A. An improved genetic algorithm for the multi level uncapacitated facility location problem. International Journal of Computers Communications & Control. 2013;8(6):845–853.

[5] Brimberg J, Mladenovic N. Solving the continuous location-allocation problem with tabu search. Studies in Locational Analysis. 1996;8(23-32):41.

[6] Murray AT, Church RL. Applying simulated annealing to location-planning models. Journal of Heuristics. 1996;2(1):31–53.

[7] Ernst AT, Krishnamoorthy M. Solution algorithms for the capacitated single allocation hub location problem. Annals of Operations Research. 1999;86:141–159.

[8] Brimberg J, Hansen P, Mladenović N, Taillard ED. Improvements and comparison of heuristics for solving the uncapacitated multisource Weber problem. Operations Research. 2000;48(3):444–460.

[9] Gong D, Gen M, Yamazaki G, Xu W. Hybrid evolutionary method for capacitated location-allocation problem. Computers & industrial engineering. 1997;33(3):577–580.

[10] Salhi S, Gamal M. A genetic algorithm based approach for the uncapacitated continuous location–allocation problem. Annals of Operations Research. 2003;123(1):203–222.

[11] McCormack R, Coates G. A simulation model to enable the optimization of ambulance fleet allocation and base station location for increased patient survival. European Journal of Operational Research. 2015;247(1):294–309.

[12] Jahantigh FF, Malmir B. A Hybrid Genetic Algorithm for Solving Facility Location Allocation Problem. In: Proceedings of the 2016 International Conference on Industrial Engineering and Operations Management, Kuala Lumpur, Malaysia; 2016. .

[13] Neema MN, Maniruzzaman KM, Ohgai A. Application of Advanced Hybrid Genetic Algorithms for Optimal Locations of High School. In: International Conference on Computational Science and Its Applications. Springer; 2013. p. 294–303.

[14] Saeidian B, Mesgari MS, Ghodousi M. Evaluation and comparison of Genetic Algorithm and Bees Algorithm for location–allocation of earthquake relief centers. International Journal of Disaster Risk Reduction. 2016;15:94–107.

[15] Stanimirović Z. A genetic algorithm approach for the capacitated single allocation p-hub median problem. Computing and Informatics. 2012;29(1):117–132.

[16] Shariff SR, Moin NH, Omar M. Location allocation modeling for healthcare facility planning in Malaysia. Computers & Industrial Engineering. 2012;62(4):1000–1010.

[17] de Oliveira RM, Mauri GR, Lorena LAN. Clustering search for the berth allocation problem. Expert Systems with Applications. 2012;39(5):5499–5505.

[18] Shankar BL, Basavarajappa S, Chen JC, Kadadevaramath RS. Location and allocation decisions for multi-echelon supply chain network–A multi-objective evolutionary approach. Expert Systems with Applications. 2013;40(2):551–562.

[19] Mousavi SM, Niaki STA, Mehdizadeh E, Tavarroth MR. The capacitated multi-facility location–allocation problem with probabilistic customer location and demand: two hybrid meta-heuristic algorithms. International Journal of Systems Science. 2013;44(10):1897–1912.

[20] Michalewicz Z. Genetic algorithms+ data structures= evolution programs. Springer Science & Business Media; 2013.

[21] Rogers A, Prugel-Bennett A. Genetic drift in genetic algorithm selection schemes. IEEE Transactions on Evolutionary Computation. 1999;3(4):298–303.

[22] Baluja S. Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning. Carnegie-Mellon Univ Pittsburgh Pa Dept Of Computer Science; 1994.

[23] Baluja S, Caruana R. Removing the genetics from the standard genetic algorithm. In: Machine Learning: Proceedings of the Twelfth International Conference; 1995. p. 38–46.

[24] Folly KA, Venayagamoorthy GK. Effects of learning rate on the performance of the population based incremental learning algorithm. In: Neural Networks, 2009. IJCNN 2009. International Joint Conference on. IEEE; 2009. p. 861–868.

[25] Baluja S. An empirical comparison of seven iterative and evolutionary function optimization heuristics. 1995;.

[26] Beasley JE. OR-Library: distributing test problems by electronic mail. Journal of the operational research society. 1990;p. 1069–1072.

[27] Filipović V. Fine-grained tournament selection operator in genetic algorithms. Computing and Informatics. 2012;22(2):143–161.