Comparison of Genetic Algorithms, Random Restart, and Two-Opt Switching for Solving Large Location-Allocation Problems

Christopher R. Houck, Jeffrey A. Joines, and Michael G. Kay†

Department of Industrial Engineering
North Carolina State University
Raleigh, NC 27695-7906

Subject classifications: Facilities: location-allocation; analysis of algorithms: genetic algorithms, random restart, and two-opt switching

† Corresponding author

Scope and Purpose—The location-allocation problem is a type of multifacility location problem in which both the location of new facilities and the allocation of existing facility requirements to the new facilities is determined. An example of a location-allocation problem involves the design of a distribution network where, in addition to determining the location of each warehouse, we also want to determine the allocation of customers to warehouses.

The location-allocation problem is a difficult optimization problem because multiple local minima may exist. Large problem instances (more than 25 existing facilities) require the use of heuristic procedures. The main purpose of this paper is to compare the performance of three heuristic procedures for solving the problem: genetic algorithms, random search, and two-opt switching. Our analysis shows that genetic algorithms provide better solutions than either of the traditional procedures, random search and two-opt switching, and with less computational effort.

Abstract—This paper examines the application of a genetic algorithm used in conjunction with a local improvement procedure for solving the location-allocation problem, a traditional multifacility location problem. This problem is difficult to solve using traditional optimization techniques because of its multimodal, nonconvex nature. The alternate location-allocation (ALA) method has been shown to be an effective local improvement procedure for the location-allocation problem. Using the ALA method, an empirical analysis was done to determine the number and size of the local minima of the location-allocation problem to demonstrate the reduction of the size of the search space that can be achieved through the use of the ALA method as an evaluator. A genetic algorithm that evaluates a series of ALA solutions was developed and compared to two traditional heuristic procedures for the problem: random restart and H4, a two-opt procedure. Like the genetic algorithm, both procedures evaluate a series of ALA solutions. A statistical analysis of the quality of the solutions provided by the three procedures for several problems of varying size demonstrated that the genetic algorithm provides the best solutions. An examination of the number of ALA evaluations performed by each procedure showed that the genetic algorithm also found solutions to the larger size problems much quicker than either the random restart or the procedures.

1. INTRODUCTION

In this paper, we consider the location-allocation problem in which both the location of n new facilities (NFs) and the allocation of the flow requirements of m existing facilities (EFs) to the NFs are determined so that total transportation costs are minimized. Given m EFs located at known points a_j, $j = 1, \ldots, m$, with associated flow requirements w_j, $j = 1, \ldots, m$, and the number of NFs, n, from which the flow requirements of the EFs are satisfied, the location-allocation problem can be formulated as the following nonlinear program:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}\, d(X_i, a_j) \\ \text{subject to} \quad & \sum_{i=1}^{n} w_{ij} = w_j, \quad j = 1, \ldots, m \\ & w_{ij} \geq 0, \quad i = 1, \ldots, n, \; j = 1, \ldots, m, \end{aligned} \tag{1}$$

where the unknown variable locations of the NFs, X_i, $i = 1, \ldots, n$, and the flows from each NF i to each EF j, w_{ij}, $i = 1, \ldots, n$, $j = 1, \ldots, m$, are the decision variables to be determined, and $d(X_i,a_j)$ is the distance between NF i and EF j. An infinite number of locations are possible for each NF i since each X_i is a point in a continuous, typically two-dimensional, space.

The location-allocation problem (1) is a difficult optimization problem because its objective function is neither convex nor concave [2], resulting in multiple local minima. Optimal solutions to the problem must lie at one of the extreme points of the constraint set of problem (1) [2]. Exact solutions to (1) are generally limited to problems with up to 25 EFs for general l_p distances [12] and up to 35 EFs for rectilinear distances [13]. However, it is not uncommon to have location-allocation problems which contain hundreds of EFs serviced by dozens of NFs [3].

One method for solving the location-allocation problem, the alternate location-allocation (ALA) method [1], quickly finds a local minimum solution given a set of starting locations for the NFs. Other heuristic procedures, which examine multiple local minima by changing the allocation of one or two of the EFs in the solution found by the ALA method, have only been used to solve problems with less than 100 EFs and up to 10 NFs [14]. In the research reported in this paper, it was found that the computational complexity of these heuristics make them impractical for solving larger problems due to the inefficiencies of the local search strategies they employ. Stochastic search strategies, such as simulated annealing and genetic algorithms, employ a global search over the solution space. These strategies have been shown to efficiently find good solutions to many problems [4, 15]. This paper examines the use of genetic algorithms, used in conjunction with a local improvement procedure, for solving the location-allocation problem and compares its performance on a series of test problems to the performance of two traditional heuristics: a random restart procedure and the two-opt heuristic, H4, presented by Love and Juel [12].

2. PREVIOUS RESEARCH

The ALA local improvement procedure was introduced by Cooper [1]. This method works by starting with a set of NF locations and then finding an optimal allocation of EFs based on those locations. The optimal NF locations for this allocation are then found. The method continues until no further allocation changes are made. Finding the optimal locations of the NFs for a given allocation corresponds to solving a continuous single-facility location problem for each NF. Determining the optimal allocations given the NFs locations corresponds to determining the NF closest to each EF. Starting from any set of NF locations, the ALA method generates a monotonically nonincreasing sequence of locations and allocations which lead to a local minimum solution to the location-allocation problem, where local minimum is defined as a set of

locations and allocations such that the locations are optimal with respect to the allocations and the allocations are optimal with respect to the locations [12].

Love and Juel [12] present a series of five heuristic procedures which search among local minima by manipulating a single solution found using the ALA local improvement procedure. The five heuristics all work by stepping from one local minimum to another. They differ by the mechanism used to manipulate the single solution and the circumstances under which they make a step to the next local minimum. The first three heuristics, H1, H2, and H3, make a single switch of the allocation of an EF. The H4 and H5 heuristics perform two-opt switching; switching the allocation of two EFs simultaneously. The H4 heuristic makes the first switch that decreases the total cost, whereas the H5 heuristic looks at all possible switches of two EFs and makes the switch which leads to the greatest reduction in the cost. Love and Juel compared these heuristics on a set of randomly generated rectilinear distance problems ranging in size from m = 12–100 and n = 2–10. For a series of small problems of up to m = 16 and n = 3, the H4 and H5 heuristics found the known global optimal solutions, and in all cases the H4 and H5 heuristics yielded identical results. Additional discussion of these heuristics can be found in Love et al. [14].

Since many local minima may exist in the optimization of nonconvex functions and global optimality is not guaranteed, a common practice is to search for many local minima by trying several different starting points and recording the best solution found [7]. The simplest method of generating starting points is random restart, a procedure where the starting location of each point to be sampled is generated randomly. Random restarting approaches, a form of stochastic search, are blind search techniques because they do not utilize any information from the previous solutions. The random restart (RR) procedure used to solve the location-allocation problem randomly generates a set of NF locations, applies the ALA local improvement procedure, and records the solution found; repeating this procedure for each trial.

3. PROBLEM CHARACTERISTICS

The use of the ALA local improvement procedure reduces the size of the solution space considerably because only local minima need to be examined. Eilon et al. [6] investigated an $m = 50$ and $n = 5$ problem using the ALA method and found that only 61 local minima exist, and Cooper [1] has stated that the region around the global optimal is flat rather than peaked.

To further examine the properties of the location-allocation problem, the ALA method was used to solve the $m = 7$ and $n = 2$ rectilinear distance problem presented in Francis et al. [7] The ALA method was started with every integer pair of NF locations in the rectangular convex hull of the EFs and the resulting local minima found after the application of the ALA method were recorded. The results are shown in Table 1. As the table shows, starting from a single set of two randomly located NFs there is a 30.7% chance that the ALA method will find the global optimum. Also, note that the use of the ALA method yields only nine distinct solutions, whereas the original size of the search space was 128 potential solutions (the 2^7 allocations). The use of the ALA method transforms the space of the problem from large shallow basins to only the local minima in those basins. This might explain why the two-opt switching of the H4 and H5 heuristics yields better results than the single switches of the H1, H2, and H3 heuristics: A two-opt switch moves further from the current local minimum, providing a greater opportunity to find a better local minimum.

To examine how the properties of the location-allocation problem change with increasing problem size, an estimate of both the number of local minima and the probability of finding the best solution was determined. For a series of seven rectilinear distance test problems, 5,000 sets of random NF locations were generated and the local minimum was found for each set using the ALA method. The smaller test problems were taken from the literature and the larger problems were generated randomly in a similar fashion as Love and Juel [12], with the EF locations as well as the weights being drawn from a uniform random number stream between 0 and 10,000. Table 2 shows the results of this investigation. Rectilinear distances were used due to the

computational ease of solving the location subproblem in the ALA method, as well as to maintain consistency with previous research.

In problems 1-5, the best solution is found during the 5,000 random restarts of the ALA method. Even for a large problem, problem 5, with $m = 100$ and $n = 10$, the probability of finding the best solution from a single random set of NF locations is approximately 1%. Therefore, it seems likely (and is shown below) that using the ALA method with random starting locations would perform well for even moderately large problems. For problems 6 and 7, much larger problems, the best solution was not found during the 5,000 random restarts of the ALA method. Therefore, the estimated probability of finding the best solution starting from random locations was estimated using the binomial distribution. The best solution was not found in 155,000 trials (the 5,000 random restarts reported in this section plus the 10 replications of the random restart procedure reported in the experimentation section). In order to achieve a 99 percent probability of 155,000 failures and no successes, a probability of success of less than 0.67×10^{-05} percent is required.

Further analysis of the locations of the NFs for the local minima given in Table 1 shows that there is a structure to this problem that can be exploited. While a solution from the ALA local improvement procedure might not be the global optimal, portions of the optimal location may be contained in the near optimal solutions. For example, four of the five best local minima contain the same location for the first NF. Thus, by combining parts of good solutions together, potentially better solutions could be found in a more efficient manner than a pure random search.

4. GENETIC ALGORITHMS

"In general, any abstract task to be accomplished can be thought of as solving a problem, which, in turn, can be perceived as a search through a space of potential solutions" [15]. Whereas traditional search techniques use characteristics of the problem to determine the next sampling point (e.g., gradients, Hessians, linearity, and continuity), stochastic search techniques make no

such assumptions. Instead, the next sampled point(s) is(are) determined based on stochastic sampling/decision rules rather than a set of deterministic decision rules.

Genetic algorithms, more intelligent stochastic search techniques than random restart, improve upon random restart by mimicking the evolutionary process through the use of a "survival of the fittest" strategy. In general, the fittest individuals of any population tend to reproduce and pass their genes on to the next generation, thus improving successive generations. However, some of the worst individuals do, by chance, survive and also reproduce. Genetic algorithms have been shown to solve linear and nonlinear problems by exploring all regions of the state space and exponentially exploiting promising areas through mutation, crossover, and selection operations applied to individuals in the population [15].

Genetic algorithms have been shown to be able to efficiently solve difficult problems that have objective functions that do not possess "nice" properties [4, 5, 8, 9, 15]. They maintain and manipulate a family, or population, of solutions in their search for better solutions. This provides an implicit as well as explicit parallelism that allows for the exploitation of several promising areas of the solution space at the same time. The genetic algorithm (GA) procedure used to solve the location-allocation problem is a nontraditional methodology that uses a real-valued representation and evolutionary procedure as developed by Michalewicz [15]. The procedure is summarized as follows:

Genetic Algorithm (GA)

Step 1. Set generation counter $i = 0$,

Step 2. Create the initial population, Pop(i), by generating N sets of n random NF locations,

Step 3. Determine the fitness of each individual (i.e., each set of NF locations) in the population by applying the ALA local improvement procedure to the individual,

Step 4. Increment to the next generation, $i = i + 1$,

Step 5. Create the new population, Pop(i), by:

(a) selecting N individuals stochastically based on the fitness from the previous population, Pop(i–1), and

(b) randomly selecting R ($R << N$) parents to produce children through the application of genetic operators,

Step 6. Evaluate the fitness of the newly formed children by applying the ALA method as in step 3,

Step 7. If i is less than the maximum number of generations to be considered, go to step 4,

Step 8. Print out the best solution found.

A more complete discussion of genetic algorithms, including extensions and related topics, can be found in the books by Davis [4, 5], Goldberg [8], Holland [9], and Michalewicz [15].

For any genetic algorithm, a chromosome representation is needed to describe each solution or individual in the population of interest. The representation scheme determines how the problem is structured and which genetic operators are used. Each individual (or chromosome) is made up of a sequence of genes from a certain alphabet. An alphabet could be binary numbers, real (floating point) numbers, integer numbers, symbols, matrices, etc. A floating point representation is used in the GA for the location-allocation problem on a two-dimensional plane with m EFs located in the rectangle (L_x, L_y)–(U_x, U_y). An individual consists of n (x, y) pairs representing the locations of the NFs, where x and y take on real values between L_x and U_x and L_y and U_y, respectively,

$$\text{Individual} \rightarrow (x_1, y_1, x_2, y_2, \ldots, x_n, y_n).$$

Once the initial population is randomly created, each individual is evaluated using an evaluation function to determine its fitness value. Evaluation functions of many forms can be used, subject to the minimal requirement that the function can map the population into a partially ordered set. The GA for solving the location-allocation problem uses the ALA method as its

evaluation function. Since the ALA method finds a local minimum solution for a given set of starting NF locations, the fitness of each individual (i.e., set of NF locations) is determined by the cost of the solution found after the application of the ALA method to the individual. Note, the ALA method is used to evaluate and to physically change an individual.

After the population (of size N) has been evaluated, a new population of size N individuals is selected from the previous generation. The selection of individuals to produce successive generations plays an extremely important role in a genetic algorithm. These individuals do not have to be distinct, that is, an individual in the population can be selected more than once. A probabilistic selection is performed where each individual is assigned a probability based upon its fitness such that the better individuals have an increased chance of being selected. However, all of the individuals in the population have a chance of being selected to reproduce into the next generation.

The normalized geometric ranking scheme [10] was used in the GA procedure described in this paper. The individuals in the population are ranked from best to worst according to their fitness value. Then, each individual is assigned a probability of selection based upon the normalized geometric distribution,

$$P[\text{Selecting } i\text{th individual}] = q'(1-q)^{i-1},$$

where $q' = q/\left(1-(1-q)^N\right)$, q is the probability of selecting the best individual, i is the rank of the individual (where 1 is the best), and N is the size of the population.

After the new population is selected, each genetic operator is applied a discrete number of times to randomly selected individuals within the new population such that the total number of children created by all of the operators equals R. Mutation and crossover are the two basic types of genetic operators [15]. Mutation operators tend to make small random changes in one parent to form one child. Crossover operators combine information from two parents to form

two offspring such that the two children contain a “likeness” (a set of building blocks) from each parent. The application of these two basic types of operators, and their derivatives, depends on the chromosome representation used. The GA used to solve the location-allocation problem employs seven genetic operators developed by Michalewicz [15] to work with a floating point representation: uniform mutation, non-uniform mutation, multi-non-uniform mutation, boundary mutation, simple crossover, arithmetic crossover, and heuristic crossover.

The “uniform mutation” operator randomly selects one of the variables, x_i or y_i, from a parent and sets it equal to a random number uniformly distributed between the variable’s lower bound, L_x or L_y, and upper bound, U_x or U_y. The “boundary mutation” operator randomly selects one of the variables from a parent and randomly sets it equal to its lower or upper bound. The “non-uniform mutation” operator randomly selects one of the variables, x_i or y_i, from a parent and sets it equal to a random number from a non-uniform distribution [15]. In early generations, this operator is similar to the uniform mutation operator, but, as the number of generations increases, the spread of the distribution narrows to zero, increasing the exploitation of the local solution. The “multi-non-uniform mutation” operator applies the non-uniform operator to all of the variables in the parent. The “simple crossover” operator randomly selects a cut point dividing each parent into two segments. The first child is created by combining the first segment from the first parent and the second segment from the second parent. The second child is created from the first segment of the second parent and the second segment of the first parent. The “arithmetic crossover” operator produces a complimentary pair of linear combinations produced from random proportions of the parents. The “heuristic crossover” operator produces a child that is a linear extrapolation away from the better parent along the direction of the vector joining the two parents.

The GA moves from generation to generation, repeating steps 4–7 until the termination criterion is met. The stopping criterion used is the specification of the maximum number of generations to iterate through. This allows one to preset the maximum number of (not

necessarily unique) solutions that are evaluated. Other termination strategies include the use of population convergence criteria and checking if there has been no improvement of the best individual over a specified number of generations. Several strategies can be used in conjunction.

5. EXPERIMENTATION

In order to evaluate the effectiveness of the genetic algorithm for solving the location-allocation problem, it was compared with the random restart (RR) procedure and the H4 heuristic of Love and Juel [12] on the same seven test problems described in Section 2. The test problems, as shown in Table 2, were selected based on their varying size and complexity. The smaller test problems, with known optimal solutions, were taken from Francis et al. [7] and Love and Juel [12] and were used to validate the stochastic search techniques. The larger test problems, without known optimal solutions, were randomly generated and were used to explore the scalability of the three approaches.

The H4 heuristic was used in the experiments because, as compared to the four other heuristics presented and evaluated by Love and Juel [12–14], it provided the best mix of solution quality and computational efficiency. H4 was used instead of H5 due to the excessive computational complexity of H5. As mentioned in Section 2, the H4 heuristic makes the first switch that decreases the total cost, whereas H5 looks at all possible switches and makes the switch which leads to the greatest reduction in the cost. This leads to the following number of ALA local improvement procedure evaluations performed by H5 at each step (i.e., before any permanent change in the current allocation is made):

$$\frac{1}{2}\sum_{i=1}^{n}(m-n_i)(m-n_i-1), \tag{2}$$

where n_i = the number EFs allocated to NF i. Even though H4 can change its permanent allocations at each step before performing this number of evaluations, it still must perform this

many evaluations before terminating on its final step. Thus, H4 and H5 both have a worst case complexity of $O(m^3)$ evaluations of the ALA method at each step. The best case performance occurs when EFs are allocated in equal proportions to the NFs (i.e., $n_i = m/n$, $i = 1, \ldots, n$); this results in the following number of evaluations:

$$\frac{1}{2}\left[m^2\left(n+\frac{1}{n}-2\right)+m(1-n)\right]. \tag{3}$$

The worst case performance occurs when all EFs are allocated to one NF (e.g., $n_1 = m$ and $n_i = 0$, $i = 2, \ldots, n$); this results in the following number of evaluations:

$$\frac{1}{2}\left[m^2(n-1)+m(1-n)\right]. \tag{4}$$

Therefore, due to the computational complexity as presented above, the H4 heuristic was chosen for comparison and modified to include the additional stopping criteria of the maximum number of potential solution evaluations.

In order to allow statistically valid comparisons to be made between the performance of the GA, RR, and H4 procedures, ten replications of each approach for each test problem were performed. Each replication was started from a set of randomly generated NF locations. Table 3 displays the various parameter values that were used with the GA for this investigation. They were selected because they have been shown to work well for a wide class of problems [15]; no attempt was made to optimize these parameter values for this particular problem. For each genetic operator, the parameter value shown in the table represents the number of children produced each generation by the operator; therefore, during each generation a total of $R = 28$ children (the sum of these parameter values) are formed.

Since the maximum number of generations was set to 100 and the size of the population is $N = 80$, the GA evaluates 2,880 not necessarily unique solutions during each replication. In the

comparison, both RR and H4 were allowed to evaluate 15,000 solutions, over five times as many evaluations as the GA. The basic H4 heuristic does not limit the number of evaluations of potential solutions. It terminates when no further two-opt switches will improve the current solution. However, to provide a computationally feasible comparison, the heuristic was terminated after evaluating 15,000 solutions. The use of H5 would have resulted in no steps being taken for the problems larger than $m = 65$ and $n = 5$, and H4 would have required far too many evaluations to terminate naturally on its final step for these problems. Therefore, the use of H5 is impractical for solving large location-allocation problems and, for H4 to be a practical approach, the number of evaluations that it considers must be limited.

6. RESULTS

The experiments were used to compare the effectiveness of the GA, RR, and H4 procedures with respect to two characteristics: the quality of the solutions they obtained and their computational efficiency. Both characteristics are important when solving large location-allocation problems because the solution space of these problems is so large that computation time becomes a limiting factor in the quality of the solutions that can be obtained.

6.1. Solution Quality

The quality of the solutions found in these experiments is shown in the box and whiskers plots of Fig. 1. The boxes have lines at the lower quartile, median, and upper quartile values, and the whiskers show the range of the rest of the data. As can be seen, the quality and variation of the solutions obtained using H4 degrade as the problem size increases, starting with a moderate size problem, problem 4. As shown in the figure, the stochastic sampling techniques, RR and GA, outperform H4 in both the quality and variance of their solutions, with the GA being superior to RR for the larger problems. However, to determine if the differences in the quality of the solutions provided by the GA, RR, and H4 were statistically significant, a two-sample *t*-test was

performed. Welch's approximation of the number of degrees of freedom was used in the test due to the great disparity among the variances of the three approaches [11, 16].

The results of these *t*-tests are presented in Table 4. Due to the small sample size, a Shapiro–Wilk test for normality was performed at an alpha level of 0.05. Those data sets failing the test are designated in Table 4. The entries in the cells correspond to the probability of accepting the null hypothesis, that is, that the means of the approaches are equal. To determine if all three approaches are simultaneously equal, the Bonferroni inequality was used [11]. The Bonferroni inequality was used instead of an ANOVA since an ANOVA assumes that the variances are equal [11]. From the box and whiskers plots of Fig. 1, it is evident that the variances among the means of the approaches are not all equal. Table 4 and the figure show that the GA and RR yield identical solutions for test problems 1–5. However, H4 produces statistically significant worse solutions than either the GA or RR for problems 4–7. For the very large problems, problems 6 and 7, the GA provides statistically significant better solutions than either RR or H4.

6.2. Computational Efficiency

The rate at which any algorithm improves its current solution is important in any time-limited application. Because of the computational requirements of the ALA method, the number of ALA evaluations required becomes a limiting factor; for example, problem 7, requires on the order of 10 seconds to perform one ALA evaluation on a DECstation 5000 and, therefore, to examine 15,000 potential solution would require 41.67 hours. To compare the computational efficiency of the GA, RR, and H4 procedures, the average over ten replications of the number of solutions examined (i.e., evaluations of the ALA local improvement procedure) before finding the final solution reported for each test problem by each approach is shown in Fig. 2. The GA always evaluates at least 80 solutions, the size of its initial randomly generated population, before reporting any results; therefore, it is equivalent to the random restart approach for its first

generation. In addition, the final solution reported by each approach over the ten replications might not be as good as the best solution found for that problem by the other approaches.

As can be seen from Fig. 2, all three approaches quickly found the known optimal solutions for the small problems, problems 1–3. On average, RR examined 4.1, 6.6, and 4.9 solutions, respectively, for these problems; H4 examined 2.7, 45.2, and 53.5 solutions, respectively; and the GA always found the optimal solutions in its initial population. However, for the medium sized problems, problems 4 and 5, RR and the GA quickly found the best known solutions (examining, on average, 16.8 and 178.4 solutions and 80 and 119.2 solutions, respectively), while H4 had to examine many more solutions before finding its final solution, which was not the best known solution for these two problems (see Fig. 1). For the larger problems, problems 6 and 7, the GA reached significantly better solutions much more quickly than either H4 or RR. Although H4 found its best solutions for these large problems more quickly than random restart, the quality of the solutions found by H4 was not as good as the quality of the solutions found by RR (see Fig. 1). Note, although RR appears to reach its best solution quicker for problem 7 than for problem 6, the difference is not statistically significant.

7. CONCLUSIONS

This paper has examined three different approaches for solving large location-allocation problems: genetic algorithm (GA), random restart (RR), and H4 heuristic procedures. Each procedure uses the ALA local improvement procedure. A series of seven test problems of varying size was used to compare the performance of the three procedures with respect to both the quality of the solution obtained by each and its computational efficiency as measured by the number of ALA evaluations required. For the smaller size test problems, all three procedures found the known optimal solutions quickly with very similar computational requirements. However, for the medium and larger size test problems, the H4 heuristic produced statistically significant worst solutions than either the RR procedure or the GA, and, for the larger size

problems, the GA found statistically significant better solutions with less computational effort than either RR or H4. The differences in the computational efficiency of the three approaches would be even more critical had general l_p distances been used in the test problems instead of rectilinear distances because iterative, as opposed to exact, procedures would have been required for the locational decisions in each ALA evaluation.

The differences in the performance of the three approaches can be explained by the characteristics of the local minima of the problem, namely, that the local minima are in large shallow basins. The fact that all of the approaches worked well for the smaller size problems can be explained by the small number of local minima that needed to be examined. For the larger size problems, the number of local minima increased dramatically; therefore, efficient global search strategies were needed to find good solutions. The H4 heuristic did not perform well on these problems because only small changes in the allocation of EFs to NFs are made at each step of the procedure. As a result of this two-opt switching strategy, H4 was only able to search a small portion of the solution space of local minima. In contrast, the stochastic global search strategies used in RR and the GA allowed local minima distributed across the entire solution space to be searched with an equivalent amount of computational effort as H4 and with much greater opportunities for finding better solutions.

The differences in the quality of the solutions found by RR and the GA for the larger size problems can be explained by the very small probability of finding the best known solutions for these problems (see Table II) using just the simple random (or blind) search strategy of random restart. The ability of the GA, through the use of genetic crossover operations, to create an improved solution by combining portions of two existing solutions enabled more promising solutions to be quickly exploited for an equivalent amount of computational effort as required by random restart for its blind search. For the smaller size problems, the probability of finding the best known solutions was so high that they were quickly found by the simple random sampling used both in random restart and in the first generation of the GA before the genetic operators

created its second generation. Thus, in general, while the GA is the most effective approach for large location-allocation problems, the use of the ALA local improvement procedure with random starting locations for the NFs (i.e., the RR procedure) is also an effective, and easier to implement, means of solving smaller problems.

Acknowledgments—The authors would like to thank Halim Damerdji, Shu-Cherng Fang, and James R. Wilson for their helpful comments and suggestions.

REFERENCES

1. L. Cooper, Location-Allocation Problems. *Opns. Res.* **11**, 331–343 (1963).
2. L. Cooper, The Transportation-Location Problems. *Opns. Res.* **20**, 94–108 (1972).
3. M.S. Daskin and P.C. Jones, A New Approach to Solving Applied Location/Allocation Problems. *Microcomputers in Civil Engineering* **8**, 409–421 (1993).
4. L. Davis, *Genetic Algorithms and Simulated Annealing*. Pitman, London (1987).
5. L. Davis, *The Handbook of Genetic Algorithms.* Van Nostrand Reingold, New York (1991).
6. S. Eilon, C.D.T. Watson-Gandy, and N. Christotides, *Distribution Management: Mathematical Modeling and Practical Analysis.* Nafner, New York (1971).
7. R.L. Francis, L.F. McGinnis, Jr. and J.A. White, *Facility Layout and Location: An Analytical Approach*, Second Edition. Prentice-Hall, Englewood Cliffs, NJ (1992).
8. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, PA (1989).
9. J.H. Holland, *Adaptation in Natural and Artificial systems.* The University of Michigan Press, Ann Arbor (1975).
10. J.A. Joines and C.R. Houck, On the Use of Non-Stationary Penalty Functions to Solve Constrained Optimization Problems with Genetic Algorithms. *Proc. IEEE Conf. on Evolutionary Computation*, Orlando, FL, 579–584, (1994).
11. A.M. Law and W.D Kelton, 1991. *Simulation Modeling and Analysis,* Second Edition. McGraw Hill, New York.

12. R.F. Love and H. Juel, Properties and Solution Methods for Large Location-Allocation Problems. *J. Opns. Res. Soc.* **33**, 443–452 (1982).

13. R.F. Love and J.G. Morris, A Computational Procedure for the Exact Solution of Location-Allocation Problems with Rectangular Distances. *Naval Res. Logist. Q.* **22**, 441–453 (1975).

14. R.F. Love, J.G. Morris and G.O. Wesolowshy, *Facilities Location: Models & Methods*. North-Holland, New York (1988).

15. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York (1992).

16. B.L. Welch, The Generalization of Student's Problem when Several Different Population Variances are Involved. *Biometrika*, **34**, 28–35 (1947).

Table 1. Local minima found using randomly located new facilities

Local Minimum	Objective Value	Est. Prob. of Finding Local Minimum	Location of New Facilities
1	18	30.7%	(0,2) (7,4)
2	21	16.0%	(2,0) (5,5)
3	21	17.5%	(0,2) (7,0)
4	21	10.0%	(0,2) (5,2)
5	22	10.4%	(2,2) (5,2)
6	23	0.8%	(2,0) (3,4)
7	24	9.3%	(0,0) (3,4)
8	24	4.6%	(0,5) (3,2)
9	24	1.0%	(3,0) (2,4)

Table 2. Probability of finding best solution and number of local minima found

Test Problem	Reference	No. of EFs (*m*)	No. of NFs (*n*)	Est. Prob. of Finding Best Solution	Est. Number of Local Minima
1	Francis [7]	7	2	30.7%	9
2	Love and Juel [12]	20	3	21.5%	10
3	Love and Juel [12]	35	2	18.8%	34
4	Randomly Generated	65	5	9.1%	>300
5	Randomly Generated	100	10	0.9%	>4000
6	Randomly Generated	250	25	0.67×10^{-5}%	unknown
7	Randomly Generated	500	40	0.67×10^{-5}%	unknown

Table 3. Values of the parameters used in the genetic algorithm

Parameter	Parameter Value
Uniform Mutation	4
Non-Uniform Mutation	4
Multi-Non-Uniform Mutation	6
Boundary Mutation	4
Simple Crossover	4
Arithmetic Crossover	4
Heuristic Crossover	2
Probability of selecting the best, q	0.08
Population Size, N	80
Maximum No. of Generations	100

Table 4. *t*-test results for GA, RR, and H4

Test Problem	GA vs. H4	GA vs. RR	RR vs. H4	Simultaneously Equal
1	—	—	—	—
2	—	—	—	—
3	0.15966†	—	0.15966†	—
4	0.00222†	—	0.00222†	—
5	0.00069	—	0.00069	—
6	3.20309×10^{-5}	2.88724×10^{-7}	7.07513×10^{-5}	1.03071×10^{-4}
7	7.82665×10^{-6}	5.28999×10^{-12}	4.05670×10^{-4}	4.13495×10^{-4}

† The H4 data sets failed the Shapiro–Wilk test for normality at an alpha = 0.05

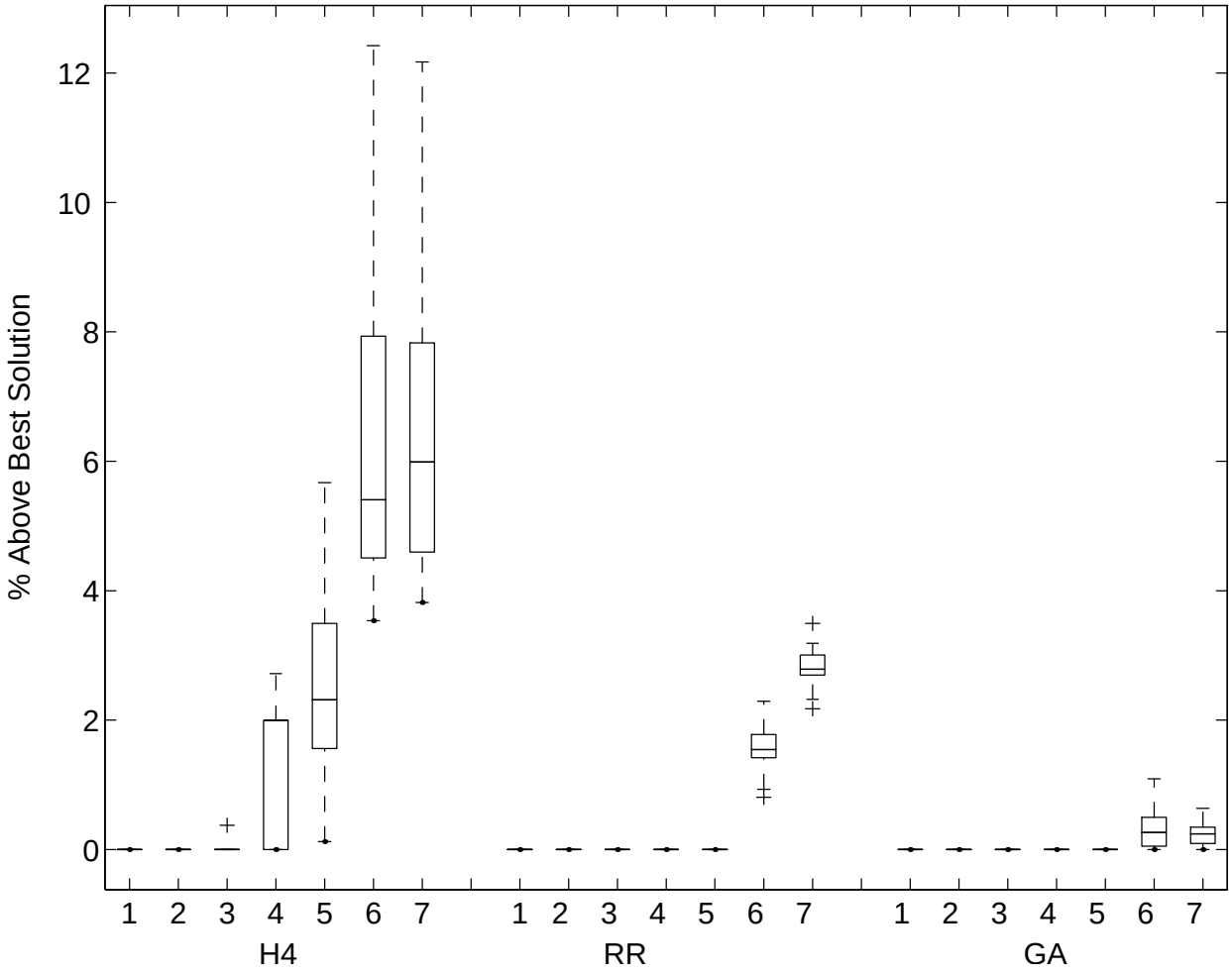

Fig. 1. Solution quality of H4, RR, and GA for problems 1–7.

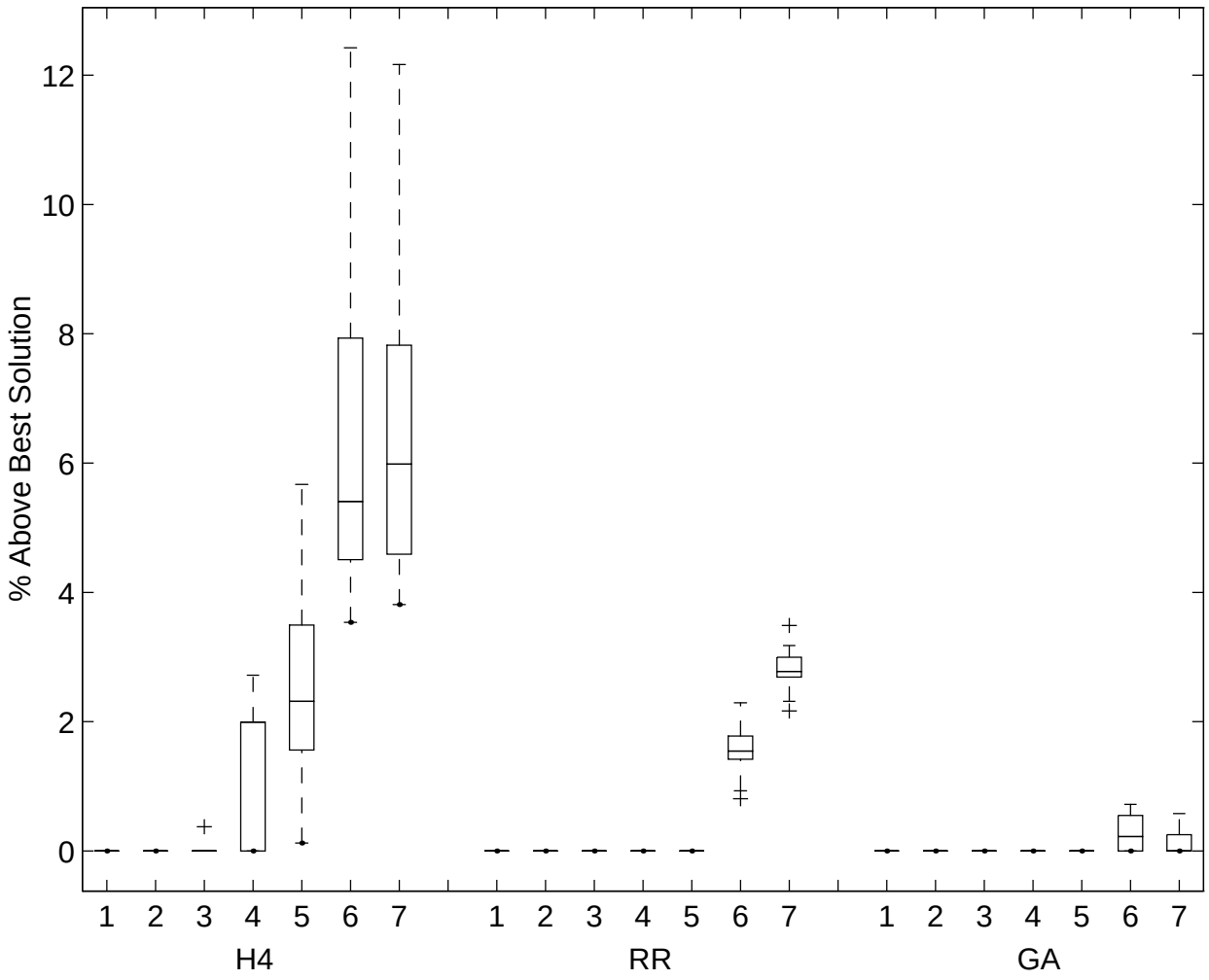

Fig. 2. Computational efficiency of H4, RR, and GA for problems 1–7.

Christopher R. Houck is a doctoral student in the Department of Industrial Engineering at North Carolina State University. He received his B.S. in industrial engineering from Penn State and M.S. in industrial engineering from NCSU. His current research interests include artificial neural networks, genetic algorithms, and the control of flexible manufacturing systems. He is a member of Phi Kappa Phi, and Alpha Pi Mu.

Jeffrey A. Joines is a doctoral student in the Department of Industrial Engineering at North Carolina State University. He received his B.S. in industrial engineering, B.S. in electrical engineering, and M.S. in industrial engineering from NCSU. His current research interests include object-oriented simulation, artificial neural networks, and genetic algorithms as applied to manufacturing. He is a member of INFORMS, IEEE, IIE, Phi Kappa Phi, and Alpha Pi Mu.

Michael G. Kay is an Assistant Professor in the Department of Industrial Engineering at North Carolina State University. His current research interests are focused in sensor-based control of free-ranging AGV systems and facilities layout and location. He received the Ph.D. degree in industrial engineering from North Carolina State University. He has served as Program Co-Chair of the 1994 IEEE International Conference on Multisensor Fusion and Integration and is a member of IEEE, IIE, and INFORMS.