Currently under second review in *European Journal of Operational Research*

CHARACTERIZING SEARCH SPACES FOR TABU SEARCH

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

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

Abstract

A large number of heuristic search algorithms are available for function optimization. Each of these heuristics, e.g., simulated annealing, genetic algorithms, tabu search, etc., has been shown to be effective at finding good solutions efficiently. However, little work has been directed at determining what are the important problem characteristics for which one algorithm is more efficient than the others. By examining two problems, the location–allocation problem and the quadratic assignment problem, characteristics of successful tabu search are illustrated. A tabu search for the location–allocation problem is described and implemented. The results of this tabu search are compared against a genetic algorithm. For the quadratic assignment problem, tabu search has been shown more effective than genetic algorithms; however, for the location–allocation problem, the genetic algorithm finds better solutions more efficiently than tabu search. To investigate what characteristics of the location–allocation problem makes it less amenable to tabu search, a comparison between the location–allocation problem and the quadratic assignment problem is performed. A comparison of the problem characteristics reveals that the location–allocation problem has very large basins of attraction around a few local optima. For tabu search to escape these minima requires a large number of iterations. Finally, a combination of both tabu search and genetic algorithms is presented for the location-allocation problem,where regions around genetically determined sample points are marked as tabu. This combination compares favorably to the genetic algorithm in terms of increased computational efficiency.

1.0 Introduction

Recently, a lot of attention has been directed at exploring the efficiency of meta-heuristics for solving hard search problems. Meta-heuristics guide the application and use of local heuristics. They are used to search the many local optima which most local heuristics find, attempting to locate the global optimum. This approach has met with a good deal of success. Martin and Otto [17] used 3-opt switching as a local optimizer and 4-opt switching as the meta-heuristic to search the local optima found by the 3-opt switches. This procedure found the best known solution for “moderate” sized TSP problems, 532 and 783 city problems.

The location–allocation problem involves both the location in a continuous plane of new facilities together with the allocation of the flow requirements of existing facilities to the new facilities. The location–allocation problem is a problem which has aspects of both continuous nonlinear function optimization and combinatorial optimization. Given the new facility locations or the existing facility allocations, the resulting subproblems are easy: given the allocation of existing facilities, one can determine the locations of the new facilities by solving an unimodal, unconstrained nonlinear optimization subproblem; given the location of the new facilities, the optimal allocations can easily be determined. Therefore, one can either search for the locations, a continuous nonlinear function problem, or the allocations, a combinatorial optimization problem.

Tabu search, a popular meta-heuristic for combinatorial problems has been found to be very effective at a wide range of combinatorial problems [7, 16, 21]. Several researchers have reported tabu search to be more efficient and able to yield better solutions than either genetic algorithms or simulated annealing [1, 2]. A tabu search for solving the location–allocation problem by searching for the optimal allocation is developed in this paper. The efficiency and quality of the final solutions are compared against a genetic algorithm.

A large number of heuristic search algorithms are available for function optimization. Each of these heuristics, e.g., simulated annealing, genetic algorithms, tabu search, etc., has been shown to be effective at finding good solutions efficiently. However, little work has been directed at determining what are the important problem characteristics for which one algorithm is more efficient than the others. .By examining the characteristics of two problems, the location–allocation problem and the quadratic assignment problem, problem characteristics for successful tabu search are illustrated. Recognize that any optimization technique can be fine tuned for a particular problem, this paper demonstrates that certain problems have characteristics which lend themselves to a particular standard meta-heuristic.

Section 2 introduces the location–allocation problem along with relevant previous research. Section 3 presents the details of the tabu-search implementation for the solution of the location–allocation problem. Section 4 examines the problem characteristics of both the location–allocation problem and the quadratic assignment problem. Owing to the analysis of this section, the concepts of Tabu search are combined with a genetic algorithm approach.

2.0 Location–Allocation Problem

In the location–allocation (LA) problem, 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, $j = 1,..., m$, with associated flow requirements w_j, $j = 1,..., m$, 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 combinatorial problem:

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} w_j d(X_i, a_j) \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, \ldots, m$$

$$x_{ij} \in [0, 1], \qquad i = 1, \ldots, n, j = 1, \ldots, m$$

where the unknown variable locations of the NFs, X_i, $i = 1,..., n$, and the allocation of the flow from each EF *j* to each NF *i*, x_{ij}, $i = 1,..., n$, $j = 1,..., m$, are the decision variables to be determined, and $d(X_i,a_j)$ is the distance between NF *i* and EF *j*. Given the allocations, x_{ij}, determination of the locations for each NF *i* on the continuous plane can be achieved by the solution of a convex nonlinear optimization problem.

The location–allocation problem (1) is a difficult optimization problem because its objective function is neither convex nor concave [5], 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) [5]. Exact solutions to (1) are generally limited to problems with up to 25 EFs for general l_p distances [13] and up to 35 EFs for rectilinear distances [14]. However, it is not uncommon to have location–allocation problems which contain hundreds of EFs serviced by dozens of NFs [6].

One method for solving the location–allocation problem, the alternate location–allocation (ALA) method [4], 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 [15]. A meta-heuristic procedure which searches for good starting NF locations for the ALA method has been used to solve much larger problems, e.g., 500 EFs and 50 NFs [11].

The ALA local improvement procedure was introduced by Cooper [4]. 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 [13].

Love and Juel [13] 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 2-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. [15].

Houck, Joines, and Kay [11] examined the location–allocation problem and applied a genetic algorithm to efficiently search for new facility locations. The GA used to solve the location–allocation problem employs a floating point representation where an individual in the population consists of $n(x,y)$ pairs (i.e., $(x_1,y_1, x_2,y_2, ..., x_n,y_n)$) representing the locations of the NFs. The GA used seven genetic operators described by Michalewicz [18] that 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 variables 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 [18]. 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 genetic algorithm uses the ALA heuristic to evaluate the NF locations generated by the genetic operators. The ALA heuristic transforms the problem from a small set of large basins of the local minima into just the local minima of those basins. The genetic algorithm was compared to the H4 heuristic and to a multistart using the ALA heuristic. The multistart algorithm for the location allocation problem is the repeated application of the ALA heuristic applied to randomly generated NF locations. The genetic algorithm was found to outperform both the H4 heuristic and multistart in terms of both solution quality and computational efficiency. Also, for moderate sized problems, up to $m = 100$ and $n = 10$, multistart was shown to consistently find the best solution, outperforming the H4 heuristic.

3.0 Tabu Search for the Location–Allocation Problem

Tabu search [9, 10] is a meta-heuristic which uses memory to guide an iterative search. At each iteration of the search, a neighborhood is examined to construct new solutions. These solutions are compared against the memory structure, i.e., tabu list, to prevent cycling. The best new solution which is not tabu is selected and the system moves to that new solution. This process continues until a predetermined termination criteria is reached, e.g., every move is tabu or a maximum number of iterations has been reached.

Tabu search has been found to be very effective for a variety of combinatorial problems [7, 16, 21]. Tabu search has been compared against simulated annealing and genetic search and found to find better solutions more efficiently for many combinatorial optimization problems [1, 2]. The location–allocation problem can be viewed as a combinatorial problem when solving for the optimal allocation vector.

A tabu search for the location–allocation problem is developed and presented below. A tabu search for the location–allocation problem requires the construction of a neighborhood, memory structure, aspiration criteria, and tabu-list length. Each of these elements of the tabu search, working on the allocation vector in a similar manner as the heuristics developed by Love et al. [15], is discussed in detail.

The neighborhood used is the set of all allocations which differ from the current allocation vector at exactly one existing facility. This is the neighborhood used in the H1–H3 heuristics. The use of this neighborhood generates $m(n - 1)$ neighboring solutions. To evaluate this neighborhood involves solving mn single facility planar location problems. In the current solution, there are exactly n distinct star-graph networks, as there is no interaction between the new facilities. All of the flow of each existing facility is completely assigned to a single new facility; therefore, each new facility and its allocated existing facilities is an independent network, corresponding to a single facility location problem. A neighboring solution involves changing the allocation of existing facility i, $1 \le i \le m$, from new facility j, $1 \le j \le n$, to new facility k, $1 \le k \le n$, $k \ne j$. The total cost of the neighboring solution can be efficiently found by determining the change in the costs of the two networks affected by this move, where for new facilities j and k, $\Delta T_{ik} = \Delta C_j + \Delta C_k$. The change in cost for the network associated with new facility j, ΔC_j, is the difference between the original cost of the network and the cost of the optimal single facility location problem for new facility j, excluding existing facility i. Similarly, ΔC_k is the difference between the original cost and the cost of the optimal single facility location problem for new facility k including existing facility i. Therefore, to evaluate the cost of all $m(n - 1)$ neighboring solutions requires the solving of mn single facility planar location problems as shown below.

Neighborhood Evaluation Algorithm

```
1.  $i \leftarrow 1$.
2.  $j \leftarrow$ the new facility to which existing facility $i$ is allocated.
3.  Determine $\Delta C_j$ by solving a single facility planar location problem for new facility $j$
    excluding existing facility $i$.
4.  For each $k$: $1 \le k \le n$, $k \ne j$,
4a.     Determine $\Delta C_k$ by solving a single facility planar location problem for new facility $k$
        including existing facility $i$.
4b.     $\Delta T_{ik} \leftarrow \Delta C_j + \Delta C_k$.
5.  $i \leftarrow i + 1$.
6.  Repeat from step 2 if $i \le m$.
7.  Return $\Delta T_{ik}$ as the change in cost associated with each of the neighbors.
```

The memory structure, i.e., tabu list, used for this problem records, for all i, j, the last iteration for which existing facility i was allocated to new facility j. The standard aspiration criteria of allowing a move if it leads to a solution better than the best found so far was used.

The determination of the appropriate tabu list length is critical. As is common, a varying tabu list size was used, where the length of the tabu list varies uniformly between the lower limit, $mS_{\min}$, and the upper limit, $mS_{\max}$. The determination of the appropriate range, [$mS_{\min}$,$mS_{\max}$.], to allow the tabu list length to vary was empirical. A set of experiments were run on a moderate sized problem, m = 100 and n = 10, to determine an appropriate set of bounds. The results of these initial investigations are shown in Table 1. The results of a t-test show that tabu search using range 5,

Table 1: Results of Different Tabu List Length Ranges

Range	$S_{\min}$	$S_{\max}$	Span	Avg. solution	Std. solution
1	0.4	0.6	0.2	6.5657e^8	2.65823e^7
2	0.8	1.2	0.4	6.40407e^8	1.53838e^7
3	1.6	2.4	0.8	6.34387e^8	2.35536e^7
4	0.8	1.0	0.2	6.43163e^8	1.80529e^7
5	1.6	1.8	0.2	6.21366e^8	1.45742e^7
6	2.2	2.4	0.2	6.51094e^8	2.85804e^7

[220,240], yields significantly better results than the results of running tabu search with ranges 1, 2, 4 and 6 at an alpha level of 0.05, and better than the results of using range 3 at alpha of 0.16. While range 5 is statistically superior, this small set of experiments was intended to only provide a rough estimate of a good range and not to determine an optimal range. Therefore, tabu range 5 was adopted for all further runs of tabu search reported here.

The above tabu search was used to solve the set of test problems described in [11]. The tabu search was allowed to run for 5 000 iterations. This maximum number of iterations was chosen to allow tabu search more computational time than the genetic algorithm. Both the genetic algorithm and the tabu search solve a single facility location problem as a base unit of computation. To compare the computational effort of both algorithms, a record of the number of single facility subproblems solved was maintained. The results are shown in Table 2. As the table indicates, tabu search, when run for 5 000 iterations, solves approximately two orders of magnitude more single facility location subproblems. While tabu search at each iteration solves the same number of single facility subproblems, the GA using the ALA procedure, in general solves a variable number of single facility subproblems. A 95% confidence interval for the number of single facility subproblems solved was constructed for the GA to provide a basis for comparison. For each problem

Table 2: Number of Single Facility Subproblems Solved

Problem instance	Tabu	GA 95% CI
20x3	3.0000e^5	[1.4690e^4, 1.5030e^4]
35x2	3.5000e^5	[1.1670e^4,1.2120e^4]
42x4	8.4000e^5	[2.2050e^4, 2.2360e^4]
65x5	1.6250e^6	[2.8997e^4, 2.9840e^4]
76x5	1.9000e^6	[3.3500e^4, 3.4140e^4]
100x10	5.0000e^6	[6.6570e^4, 6.7612e^4]
200x20	2.0000e^7	[1.4524e^5, 1.4749e^5]
250x25	3.1250e^7	[1.9342e^5, 1.9575e^5]

instance, the number of single facility location subproblems solved by the GA passed the Shapiro-Wilk test for normality at an alpha level of 0.1.

The results of the tabu search were compared to the genetic algorithm operating on the new facility locations as described in [11]. The results of these experiments are given in Table 3.The last column of Table 3 indicates the alpha level of significance of not rejecting the hypothesis that the means of the solutions found by both the GA and tabu search are equal. The solutions found by both the GA and tabu search were tested for normality using the Shapiro-Wilk test. The problem instances failing the test at alpha level of 0.1 are indicated in the table. As Table 3 shows, the GA finds significantly better solutions for the larger problems, i.e., greater than m = 76 and n = 5, while for smaller problems both tabu search and the GA found the same solution in every replication. Tabu search for the location–allocation finds significantly worse solutions, even when allowed more computational time for searching. To examine why tabu search performs worse than GAs for the location–allocation problem, a study of the characteristics of the problem was undertaken.

Table 3: Performance of Tabu Search and Genetic Algorithms

Problem instance	Tabu search		GA		alpha H_o: $\overline{GA} = \overline{Tabu}$
	Avg.	Std.	Avg.	Std.	
35x2	$5.981e^8$	$0.000e^0$	$5.981e^8$	$0.000e^0$	—
20x3	$1.402e^8$	$0.000e^0$	$1.402e^8$	$0.000e^0$	—
42x4	$4.904e^8$	$1.666e^6$	$4.896e^8$	$0.000e^0$	0.178288†
65x5	$5.758e^8$	$4.998e^6$	$5.705e^8$	$0.000e^0$	0.008011†
76x5	$8.432e^8$	$0.000e^0$	$8.432e^8$	$0.000e^0$	—
100x10	$6.214e^8$	$1.457e^7$	$6.116e^8$	$0.000e^0$	0.059137
200x20	$7.900e^8$	$1.820e^8$	$7.604e^8$	$0.000e^0$	0.000443
250x25	$9.419e^8$	$2.333e^8$	$8.944e^8$	$2.147e^6$	0.000076

† The solutions found by tabu search failed the Shapiro-Wilk test for normality at alpha = 0.1.

4.0 Problem Characteristics

Although the tabu search performed significantly worse than the genetic algorithm (as described in [11]) for the location–allocation (LA) problem. For other combinatorial problems, tabu search has been shown to be more efficient than genetic algorithms [1, 2]. To determine why tabu search has difficulties with the LA problem, the characteristics of the location–allocation problem were compared to a problem where tabu search has been shown to be more efficient than genetic algorithms, the quadratic-assignment problem (QAP) [19].

Table 4 lists the characteristics of the QAP and the LA problem. Given a *n*-element QAP and a *m*-EF and *n*-NF LA problem, the table lists the search space size, neighborhood size, maximum number of transitions, and maximum number of local minima. The search space size is the number of possible solutions. For the QAP, the search space size is equal to the number of permutations of *n* elements; for the LA problem, the search space size is equal to Stirling's number of the second kind [8], which represents the number of ways to partition *m* elements (EFs) to *n* blocks (NFs) assuming each block has at least one element (i.e., each NF is assigned to at least one EF).

The neighborhood size is the number of solutions that are considered at each step of the local improvement procedure used for each problem. For the QAP, the neighborhood size is the number of pairwise interchanges considered from the current solution; for the LA problem, it is the number of different NFs that can be assigned to each EF times the number of EFs, where one NF is implicitly determined by the assignment of the other NFs. The maximum number of transitions represents the maximum number of steps required to transform any starting solution into any other possible solution. For the QAP, the maximum number of transitions is equal to the maximum number of pairwise interchanges (i.e., transpositions) required to transform one permutation to another; for the LA problem, it is equal to a change of the NF assigned to each EF, where, since all of the NFs are identical, one assignment in any pair of solutions is always correct. The maximum number of local minima is equal to number of disjoint neighborhoods necessary to cover the entire search space, and, for each problem, is determined by dividing the search space size of the problem by the neighborhood size.

Table 4: Comparison of Characteristics for QAP and Location–Allocation

Characteristic	QAP	Location–Allocation
Search space size	$n!$	$\left\{ {m \atop n} \right\} = \sum_{k=0}^{n} \frac{(-1)^k (n-k)^m}{k!\,(n-k)!}$
Neighborhood size	$\binom{n}{2} = \frac{n(n-1)}{2}$	$m(n-1)$
Max. no. transitions	$n-1$	$m-1$
Max. no. of local minima	$\frac{n!}{\binom{n}{2}}$	$\frac{\left\{ {m \atop n} \right\}}{m-1}$

In addition to the maximum possible number of local minima as given in Table 4, an estimate of the actual number of local minima can be determined through random sampling. Let $A(x) = x_i^*$, where A is the local optimization function which when applied to a point x leads to the local optima x_i^*. The share (or relative region of attraction) of x_i^* is denoted θ_i. A number N of random solutions $\{x_1, x_2, \ldots, x_N\}$ is generated and the local optimization procedure A is applied to each.

Let N_i be the number of points x_j such that $(x_j) = x_i^*$, i.e., the number of times local minimum i is found. Then if there are l local minima, $\sum_{i=1}^{l} \theta_i = 1$, $\sum_{i=1}^{l} N_i = N$, and the random vector $(N_1, N_2, \ldots, N_l)$ follows the multinomial distribution [22]

$$\Pr\{N_1 = n_1, N_2 = n_2, \ldots, N_l = n_l\} = \binom{N}{n_1, \ldots, n_l} \theta_1^{n_1} \ldots \theta_l^{n_l}$$

A Bayesian estimate of the number of local minima, l, as well as the relative size of the region of attraction of each local minima, θ_i, is given in [20]. The posterior probability that there are K local minima is

$$\frac{(K-1)!K!(N-1)!(N-2)!}{(N+K-1)!(K-w)!w!(w-1)!(N-w-2)!}$$

where w different local minima are found as a result of the N searches. The posterior expectation of the number of local minima is $\frac{w(N-1)}{N-w-2}$, and the posterior expected relative size of the non-observed regions of attraction is $\frac{w(w+1)}{N(N-1)}$.

At the *i*th transition of a heuristic using a neighborhood function, the number of new solutions examined is given by the neighborhood size minus one for the previous solution examined at the (i – 1)th transition. Therefore, in *k* transitions the number of solutions examined is

$$\sum_{i=1}^{k} (im(n-1) - i) = (m(n-1) - 1)\frac{k(k+1)}{2} \qquad (2)$$

for the location–allocation problem and

$$\sum_{i=1}^{k} \left(i\binom{n}{2} - i\right) = \left(\binom{n}{2} - 1\right)\frac{k(k+1)}{2} \qquad (3)$$

for the QAP. The average share of a local minima is the search space size divided by the posterior expectation of the number of local minima. Therefore, on average, for tabu search to completely map out the basin of attraction of a local minima would require the neighborhood function to examine every solution in the basin. Specifically, by setting Eq. (2) (resp. Eq. (3)) equal to the average share and solving for *k* provides an estimate of the number of transitions required to completely map the basin of attraction for the LA problem (resp. QAP).

The characteristics of several instances of both the LA problem and QAP were analyzed using the formulae given in Table 4 and, in addition, random sampling was used to estimate both the number of local minima and *k*, the average number of transitions to map a basin. The LA instances were taken from [11], or generated in a similar manner. The QAP instances are standard problem instances taken from Burkard's QAP library [3]. The results are shown below in Table 5.

The first column identifies the problem instance and parameters, while the next four columns correspond to the characteristics described in Table 4. The observed number of local minima in Table 5 is the number of distinct local optima found during the course of the random sampling of the search space as described above. For the QAP instances, it is possible for many different solutions to be zero cost neighbors; this occurs when a pairwise interchange yields a solution with the same functional value. For this investigation, all zero cost neighbors of a local minimum were considered to be the same local minimum. Therefore, each minimum was examined to see if a series of zero cost transitions would yield an already discovered local minimum. The number of samples (No. samp.) is the number of random starting solutions, *N*, generated for the search. The estimated number of local minima is the Bayesian estimate of the number of local minima as described above. The average number of steps to completely map the basin of attraction (Avg. map steps) is provided in the last column.

Table 5 shows that as the dimensionality of the problem instances increases, the size of the search space grows rapidly; e.g., LA 42x4 is over 20 orders of magnitude larger than the LA 20x3. However, the size of the neighborhood increases at a modest rate. Also, it is interesting to note that the maximum number of transitions required to move from any solution to any other remains a relatively small number even for the largest problem instance. It is also interesting to observe that, while the maximum possible number of local minima grows rapidly with problem instance size, it appears that the actual number of local minima for these problem instances grows much more slowly.

A comparison of the characteristics of the QAP and the LA problem reveals interesting insights into why tabu search is successful for the QAP but not the LA problem. The QAP and LA problem can be compared on the basis of equal sized search spaces, an equal number of solutions considered by their respective neighborhood functions, or an equal number of maximum transitions required to move to any other solution. Examples of the results of these comparisons are provided in Table 6.

Table 5: Problem Characteristics

Problem instance	Search space size	Nbr. size	Max. no. trans.	Max. no. local min.	Obs. no. local min.	No. samp.	Est. no. local min.	Avg. map steps *k*
LA 7x2	63	7	6	9	9	10 000	9	1.11
LA 35x2	$1.718e^{10}$	35	34	$4.909e^{8}$	10	10 000	10.01	$1.0\ 1e^{3}$
LA 20x3	$5.806e^{8}$	40	19	$1.452e^{7}$	37	10 000	37.14	894.87
LA 42x4	$8.059e^{23}$	126	41	$6.396e^{21}$	525	10 000	554.15	$4.82e^{9}$
LA 65x5	$2.259e^{43}$	260	64	$8.689e^{40}$	348	10 000	360.59	$2.20e^{20}$
LA 76x5	$1.103e^{51}$	304	75	$3.628e^{48}$	888	10 000	974.66	$8.70e^{22}$
LA 100x10	$2.755e^{93}$	900	99	$3.061e^{90}$	65 302	150 000	115 650	$7.28e^{42}$
LA 200x20	$6.600e^{241}$	3800	199	$1.737e^{238}$	34 473	38 000	371 610	$3.06e^{116}$
LA 250x25	$1.968e^{324}$	6000	249	$3.280e^{320}$	30 000	30 000	†	†
QAP scr12	$4.790e^{8}$	66	11	$7.258e^{6}$	1 303	25 000	1 374.7	103.04
QAP rou12	$4.790e^{8}$	66	11	$7.258e^{6}$	2 497	25 000	2 774.2	72.39
QAP chr12a	$4.790e^{8}$	66	11	$7.258e^{6}$	1 684	25 000	1 805.7	89.85
QAP nug12	$4.790e^{8}$	66	11	$7.258e^{6}$	1 940	25 000	2 103.3	83.21
QAP scr15	$1.308e^{12}$	105	14	$1.245e^{10}$	4 100	25 000	4 904.6	220.44
QAP rou15	$1.308e^{12}$	105	14	$1.245e^{10}$	14 391	25 000	33 917	83.52
QAP chr15a	$1.308e^{12}$	105	14	$1.245e^{10}$	13 249	25 000	28 191	91.66
QAP nug15	$1.308e^{12}$	105	14	$1.245e^{10}$	6 529	25 000	8 837.4	164.10
QAP chr22a	$1.124e^{21}$	231	21	$4.866e^{18}$	49 902	50 000	$2.599e^{7}$	$6.13e^{5}$
QAP chr22b	$1.124e^{21}$	231	21	$4.866e^{18}$	48 241	50 000	$1.373e^{6}$	$2.67e^{10}$
QAP ste36a	$3.720e^{41}$	630	35	$5.905e^{38}$	29 749	29 750	†	†
† - No results after 10 000 CPU minutes of sampling on a 133MHz 604 Power PC Workstation								

Table 6: Comparison of Equal Sized QAP and LA Problem Instances

Problem instance	Problem-size criteria	Criteria value	Est. no. local min.	Avg. map steps k
QAP scr12 LA 20x3	Equal Search Space Size	$4.790e^8$ $5.806e^8$	1 303 37.14	103.04 894.87
QAP scr15 LA 42x4	Equal Nbr. Size	105 126	4 904.6 554.15	220.4 $4.82e^9$
QAP scr15 LA 20x3	Equal No. Max. Transitions	14 19	4 904.6 37.14	220.4 894.87

As the examples in Table 6 indicate, the LA problem has far fewer local minima as compared to an equal sized QAP, regardless of which of the three problem-size criteria are considered. This corresponds to the LA problem instances having very large basins of attraction about the local minima. As indicated in Table 5, the average number of iterations for tabu search to map these basins quickly grows very large. This is consistent with the results presented in Table 3. Tabu search is unable to escape from the large local minima, while the GA, with its strong random search component is able to easily jump between local minima. However, for the QAP, where there are a relatively larger number of local minima, tabu search is able to quickly map out the basin of attraction and move to a neighboring local minima.

5.0 Genetic Algorithm with Tabu Regions

Based upon the information gained from the above analysis, Tabu search is an effective search mechanism for the QAP problem but poor for large instances of the Location-Allocation problem. This is due in part to the problem's large basins of attraction. Tabu Search was developed primarily to prevent repeatedly searching the same region or cycling between local minima by maintaining a simple map of regions already searched to date. The Location-Allocation problem is different from the QAP in the that it has both a combinatorial and continuous aspect. As described in Section 2, the GA, searching the continuous space using the ALA local improvement procedure, has been shown to be more effective for this problem than a multistart approach utilizing the same procedure and the tabu search approach presented in this paper.

One of the advantages of GAs is they have been shown to exponentially exploit promising regions of the search space. Owing to their implicit parallelism (i.e., the population of solutions), the GA may exponentially exploit several regions. This benefit may cause a problem when combining local improvement procecdures with GAs. As the GA is exponentially exploiting a promising region, it may be finding solutions in the same basin of attraction when applying the local improvement.procedure. Therefore, the GA may waste a lot of computational time rediscovering the same local minima. Especially for the location-allocation problem, which has very large basins around each of the local minimum, starting the alternate location-allocation heuristic near a point which has already been examined will most likely result in the rediscovery of the same local minimum.

In order to prevent this phenomena from occurring, Tabu regions have been added to the GA to reduce the amount of wasted computation owing to the rediscovery of the same local minimum. Because the GA is working in the continuous domain, the Tabu list structure used as short term memory for combinatorial problems (e.g., the list of the past allocations, last pair-wise switches, etc.) will not be an appropriate memory structure. Instead, using the concepts of global optimization [20], hyperspheriods are placed around the already examined points, thus marking a portion of the search space as tabu.

The genetic algorithm was modified to include the concepts of tabu regions by modifying the way the genetic algorithm evaluates new solutions using the local improvement procedure. The evaluation routine developed uses a criteria taken from the global optimization literature, specifically the random linkage algorithm[12], to determine if a local search is appropriate. The details of this evaluation function is presented below

Tabu regions are incorporated into the genetic algorithm by keeping a list of points already generated by the GA to date; this list is allowed to grow indefinitely. Every time the genetic algorithm generates a new point, x_k, it first determines the distance, δ, to the nearest point already generated (i.e., the nearest point on the list). The newly generated point, x_k, is placed on the list only if the distance, δ, is greater than zero, which avoids placing duplicated points onto the list. The algorithm then determines whether or not a local search should be performed based on this distance. If a local search is deemed appropriate, the ALA heuristic is run, and the resulting local minimum is returned to the GA. Otherwise, no local improvement is conducted, and the new point x_k, and the objective function value at this point are returned to the GA

To determine when to perform a local search (i.e., an appropriate distance δ), the following measure, $\varphi_k(\delta)$, depicted in Eq (4) is used. $\varphi_k(\delta)$ provides the probability of starting a local search based on the distance to the nearest point already generated, δ. The tabu regions or hyperspheres, α_k, taken from [12], are used by to determine if a local search is necessary.

$$\varphi_k(\delta) = \begin{cases} 0 & \delta \le 0.5\alpha_k \\ \dfrac{\delta - 0.5\alpha_k}{\alpha_k} & 0.5\alpha_k \le \delta \le 1.5\alpha_k \\ 1 & \delta \ge 1.5\alpha_k \end{cases} \quad (4)$$

where

$$\alpha_k = \pi^{-1/2}\left(\Gamma\left(1 + \frac{d}{2}\right)\mu(\Omega)\,\sigma\frac{\log k}{k}\right)^{1/d}$$
$$\sigma = 10000$$
$$d = \text{Dimensionality of the problem}$$
$$\mu(\Omega) = \text{Lebesgue measure of the search space, e.g., vol}(\Omega)\text{ in}\,\mathfrak{R}^n$$

All points within a radius of $1/2\alpha_k$ are marked tabu, where k is the total number of search points generated by the genetic algorithm to date, all points from $1/2\alpha_k$ to $3/2\alpha_k$ result in a probability of conducting a local search. This yields a gradual boundary around previously explored points. This provides a criteria for determining whether or not to perform a local search in order to prevent starting local searches too close to points already examined.

This measure yields a series of monotonically decreasing hyperspheres(α_k) placed around each examined point. Therefore, the size of the tabu regions around each of these already examined points decreases as the search progresses (i.e., this provides a short term memory similar to traditional Tabu search). This also overcomes the problem of unknown minima sizes, ensuring, that smaller minima can be located as the search progresses. While at first the emphasis of the search process is on diversity, due to the large hyperspheres preventing much local search, as the search continues, α_k decreases so that, even though the probability of repeating a search increases, the probability of missing an unexplored local minimum decreases.

The GA with tabu regions (GA-Tabu) was run on the LA 200x20 and LA 250x25 location-allocation test problem instances, instances where pure tabu-search had problems locating the optimal solution. The results of the mean number of function evaluations over 30 replications are shown below in Table 7, where a function evaluation is considered to be either the solving of the location subproblem, or the solving of the allocation subproblem.

Table 7: Number of function evaluations

Problem Instance	GA	GA-Tabu	alpha H_o: $\overline{\text{GA}} = \overline{\text{GA-Tabu}}$
LA200	20 523	16 856	0.0595
LA250	73 518	47 886	0.4138

The table shows that for these larger location-allocation problem instances, the addition of tabu regions around already explored points yields quicker convergence to the best known solution. For the LA 200x20 instance, the addition of the tabu-regions yields significantly quicker convergence. However, for the LA 250x25 instance, while the difference in the means is large, there is a high degree of variability in the number of function evaluations required to locate the best known solution for this test problem instance. Therefore, the significance of the difference between the two means is low.

6.0 Conclusions

Tabu search provides a mechanism for systematically mapping out basins of attraction for local minima, thereby allowing the search to move to neighboring local minima. For problems with a large number of small basins, tabu search is an efficient means of exploring the local minima. However, for problems with few local minima, each with a very large basin of attraction, tabu search is unable to map the basin and is therefore unable to escape the local minimum to explore neighboring minima. For these problems, randomized search can be expected to perform very well.

An investigation of two problems, the location–allocation (LA) problem and the quadratic assignment problem (QAP) show that the LA problem has a small number of local minima with very large basins of attraction while the QAP has a large number of local minima with small basins of attraction. As expected, randomized search and a variant of randomized search, genetic algorithms, yield very good solutions for the LA problem, while tabu search performs relatively poorly.

The properites of tabu search (i.e., memory of the search to date) was combined with genetic algorithms and tested on two location-allocation problems. The combination algorithm performed favorably, as compared to the genetic algorithm, in terms of computational efficiency.

References

[1] E. Aarts, P. van Laarhoven, J. Lenstra, and N. Ulder. A computational study of local search algorithms for job shop scheduling. *ORSA Journal on Computing*, 6(2):118–125, 1994.

[2] R. Battiti and G. Tecchiolli. Simulated annealing and tabu search in the long run: A comparision on QAP tasks. *Computers and Mathematical Apllications*, 1994.

[3] R. Burkard, S. Karisch, and F. Rendl. QAPLIB-a quadratic assignment problem library. *EJOR*, 55:115–119, 1991.

[4] L. Cooper. Location-allocation problems. *Operations Research*, 11:331–343, 1963.

[5] L. Cooper. The transportation-location problems. *Operations Research*, 20:94–108, 1972.

[6] M. S. Daskin and P. C. Jones. A new approach to solving applied location/allocation problems. *Microcomputers in Civil Engineering*, 8:409–421, 1993.

[7] M. Dell’Amico and M. Trubian. Applying tabu search to the job-shop scheduling problem. *Annals of Operation Research*, 41:231–252, 1993.

[8] R. Francis and J. White. *Facility Layout and Location: An Analytical Approach*. Prentic-Hall, Englewood Cliffs, NJ, 1974.

[9] F. Glover. Tabu search - part I. *ORSA Journal of Computing*, 3:190–206, 1989.

[10] F. Glover. Tabu search - part II. *ORSA Journal of Computing*, 1:4–32, 1990.

[11] C. R. Houck, J. A. Joines, and M. G. Kay. Comparison of genetic algorithms, random restart, and two-opt switching for solving large location-allocation problems. *Computers and Operations Research*, 23(6), 1996.

[12] M. Locatelli and F. Schoen. Random linkage: a family of accptance/rejection algorithms for global optimization. Technical Report 156-96, University of Milano, 1996.

[13] R. Love and H. Juel. Properties and solution methods for large location-allocation problems with rectangular distances. *Journal Operations Research Society*, 33:443–452, 1982.

[14] R. Love and J. Morris. A computational procedure for the exact solution of location-allocation problems with rectangular distances. *Naval Research Logistics Quarterly*, 22:441–453, 1975.

[15] R. Love, J. G. Morris, and G. O. Wesolowshy. *Facilities Location: Models & Methods*. North-Holland, New York, 1988.

[16] M. Malek, M. Guruswamy, P. M., and O. H. Serial and parallel simulated annealing and tabu search algorithms for the travelling salesman problem. *Annals of Operations Research*, 21:59–84, 1989.

[17] O. Martin and S. Otto. Combining simulated annealing with local search heuristics. *Annals of OR*, 62, 1996.

[18] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolutionary Programs*. Springer-Verlang, New York, 1st edition, 1992.

[19] V. Nissen. Solving the quadratic assignment problem with clues from nature. *IEEE Transactions on neural networks*, 5(1):66–72, 1994.

[20] A. Rinnooy Kay and G. Timmer. Stochastic global optimization methods part I: Clustering methods. *Mathematical Programming*, 39:27–56, 1987.

[21] E. Taillard. Taboo search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 1991.

[22] A. A. Zhigljavsky. *Theory of Global Random Search*, volume 65 of *Mathematics and Its Applications (Soviet Series)*. Kluwer Academic Publishers, 1991.