# The Enhanced Evolutionary Tabu Search and Its Application to the Quadratic Assignment Problem

John F McLoughlin III
Penn State Great Valley
30 East Swedesford Road
Malvern, PA 19355, USA
1-610-531-3064
john.mcloughlin@lmco.com

Walter Cedeño
Johnson & Johnson Pharmaceutical R&D
665 Stockton Drive
Exton, PA 19341 USA
1-610-458-5264
wcedeno@acm.org

## ABSTRACT
We describe the Enhanced Evolutionary Tabu Search (EE-TS) local search technique. The EE-TS metaheuristic technique combines Reactive Tabu Search with evolutionary computing elements proven to work well in multimodal search spaces. An initial set of solutions is generated using a stochastic heuristic operator based on Restricted Candidate List. Reactive Tabu Search is augmented with selection and recombination operators that preserve common traits between solutions while maintaining a diverse set of good solutions. EE-TS performance is applied to the Quadratic Assignment Problem using problem instances from the QAPLIB. The results show that EE-TS compares favorably against other known techniques. In most cases, EE-TS was able to find the known optimal solutions in fewer iterations. We conclude by describing the main benefits and limitations of EE-TS.

## Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *heuristics.*

G.1.6 [**Mathematics of Computing**]: Optimization – *constrained optimization.*

## General Terms
Algorithms, Performance.

## Keywords
Tabu Search, Quadratic Assignment Problem, Optimization, Evolutionary Algorithms, Genetic Algorithms, Soft Computing.

## 1. INTRODUCTION
This work introduces the Enhanced Evolutionary Tabu Search (EE-TS), a metaheuristic technique that combines Reactive Tabu Search (RE-TS) [1] with evolutionary computing elements that have proven to work well in multimodal search spaces. EE-TS is a metaheuristic search technique that can be classified as a stochastic method, one of many soft computing techniques [2]. In this paper we describe the background and design of EE-TS and its performance when applied to the Quadratic Assignment Problem (QAP) [3].


*GECCO'05*, June 25-29, 2005, Washington, DC, USA.


EE-TS is initialized with a set of solutions generated using a stochastic heuristic operator based on Restricted Candidate List (RCL) [4]. RE-TS is augmented with selection and recombination operators that preserve common traits between solutions while maintaining a diverse set of good solutions. The performance of EE-TS is evaluated using problem instances from the QAP Library (QAPLIB) [5].

The QAP is NP-hard [6] and many practical instances come from areas such as design and resource allocation. The QAP is a non-trivial combinatorial optimization problem for even the small problem sizes. It deals with identifying optimal assignments of facilities to locations such that the cost of the resulting system is minimized. The QAP has application to a wide range of situations and domains, such as microprocessor design, machine scheduling, and even the topographical layout of wards and services in a hospital. Due to the complexity of problems, solutions to the QAP are often found through the application of metaheuristic search techniques, and the QAP has become in some ways a benchmark by which new techniques are validated. EE-TS is shown to compare favorably to other known techniques using a set of problems from the QAPLIB. In most cases, EE-TS was able to find the known optimal solutions in fewer iterations.

The next section presents a formal definition of the QAP, including its mathematical representation. Section 3 provides a brief overview of the tabu search. Section 4 describes the evolutionary computing enhancements in EE-TS. Section 5 provides a detailed description of EE-TS. Section 6 describes the environment used to test the performance of EE-TS. Section 7 presents the results for various problems in the QAPLIB. We conclude with a discussion of the benefits, limitations, and future enhancements of EE-TS.

## 2. The Quadratic Assignment Problem
The QAP is a resource allocation problem where the goal is to discover the best cost-effective distribution of resources. Given *N* facilities and *N* locations, assign each facility to a location such that each location is assigned a single facility and the total shipment cost is minimized. The total shipment cost is a function of the distances between the locations and the shipment costs

between the facilities and is represented by the formula shown in Equation (1),

$$f(\phi) = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} b_{\phi(i)\phi(j)} \quad (1)$$

where $a_{ij}$ represents the distance between locations $i$ and $j$, and $b_{\phi(i)\phi(j)}$ represents the shipment cost between the facilities assigned to locations $i$ and $j$.

Even with today's best computers, relatively small problems ($N=25$) require prohibitive amounts of time to solve to provable optimality [7]. Consequently, metaheuristic algorithms are commonly applied to the QAP. A summary of recent advances for solving the QAP can be found in [7].

The problem instances used in the development and testing of EE-TS were obtained from the QAPLIB, a library of QAP problems and their best-known solutions. The problems are organized into sets, with each set named after the author(s) who developed the group of problems. Some problems were generated randomly, while others are based on real-world data. The selected data sets include the Taillard set [8], the Nugent, Vollman, and Ruml set [3], and the Skorin-Kapov set [9]. Several of the selected data sets were previously used to test RE-TS [10].

## 3. Tabu Search Overview

Glover introduced Tabu search (TS) in the late 80's [11][12] [13]. The basic idea behind TS is that adding short-term memory to a local search improves its ability to locate optimal solutions. Revisiting previously or recently visited solutions is discouraged, and operations that would do so are labeled as being "tabu" or "taboo". Glover proposed the use of both statically- and dynamically-sized memory structures for tracking tabu operations. In 1991 Taillard created the Robust Tabu Search (RO-TS), which introduced a dynamic randomly-sized short-term memory design. Battiti and Tecchiolli developed the RE-TS [1] in 1994 which based the dynamic size of its short-term memory on runtime characteristics of the algorithm and which also utilized a form of long-term memory that helped prevent searches from stagnating. Many other Tabu Search variations have been developed that incorporate various forms of dynamically-sized short-term memory and long-term memory [14][15], but the RO-TS and RE-TS remain among the most successful and popular. Other approaches have been developed through experimentation with features such as socialization and competition [16] or, like the EE-TS, the integration of evolutionary operators useful for multimodal optimization. The following concepts are common to most (if not all) Tabu Search techniques, but their specific implementations are somewhat flexible.

A move is an operation by which one solution transitions into a neighboring solution. A solution's neighborhood, $N(i,k)$, is the set of all solutions that can be transitioned to from the given solution $i$ at iteration $k$ by applying a valid move. For the QAP, a common move strategy consists of swapping facilities assigned to two locations.

The Tabu List is perhaps the most influential piece of any Tabu Search design. The basic purpose of the list is to maintain a record of which moves are tabu during each iteration. Many subtleties in how this task is carried out have been shown to greatly impact the performance of Tabu Search. Usually, a move added to the Tabu List is the reciprocal of the move last accepted and applied to the current solution. The reciprocal is recorded to prevent the search from "undoing" recent moves. Several approaches exist for handling the determination of Tabu List length, but the most common are the approaches used in the Strict Tabu Search (S-TS) [12], the RO-TS, and the RE-TS.

In Battiti and Tecchiolli's RE-TS application to the QAP, the tabu list keeps track of the assignment history of each facility to each location. If the current solution has facility F1 located at location L1 and facility F2 located at location L2 and a move defined as swapping the facilities at L1 and L2 is accepted, then any move which places facility F1 back at location L1 or facility F2 back at location L2 is tabu. Just how long such a move is considered tabu is based on the length of the Tabu List.

The simplest Tabu Searches used a fixed-length list. Other techniques incorporate a dynamically changing list length throughout the course of a run. In the RO-TS, this is accomplished by randomly choosing a new list length at set intervals. The goal is to emphasize the exploration (diversification) and exploitation (intensification) of the search space. When the list length is long, it contains many tabu moves and therefore the search will be forced into new areas and directions—forced to explore. When the list is short, a fewer number of moves are tabu and the search can stay focused on solutions in a relatively small area of the search space—exploiting the smaller area.

Battiti and Tecchiolli agreed that Taillard's implementation of a dynamically-changing list size was powerful, but based the size changes in the RE-TS on dynamic characteristics of the current run [1]. Instead of the randomness of the RO-TS, the RE-TS determines whether the list length should be increased or decreased by tracking the number of duplicate solutions visited during each interval. An interval is defined as a pre-determined number of iterations within a search. If a large number of duplicates are visited, the list length is increased in order to force the exploration of other areas. If few or no duplicates are encountered, the list length is decreased to focus the search; to exploit the current area before moving on. Tracking duplicate visitations requires long-term memory sometimes referred to as frequency memory. It has generally been shown that TS techniques that include a long-term memory tend to perform better than those that do not [15].

Strict Tabu Search (S-TS) is the most straightforward technique. With S-TS all previously-visited solutions are tabu for the remainder of the run. With this approach the length of the Tabu List is always equal to the current iteration and is therefore constantly growing.

Step 1. Create an initial solution $i$ at random. Set $i^*=i$ and $k=0$.

Step 2. Set $k=k+1$ and generate a subset $V^*$ of solutions in $N(i,k)$ such that either one of the tabu conditions $tr(i,m) \in$ Tr is violated ($r=1,...,t$) or at least one of the aspiration conditions $ar(i,m) \in Ar(i,m)$ holds ($r=1,...,a$).

Step 3. Choose a best $j=i \oplus m$ in $V^*$ (with respect to objective function f) and set $i=j$.

Step 4. If $f(i) < f(i^*)$ then set $i^*=i$.

Step 5. Update tabu and apsiration conditions.

Step 6. If a stopping condition is met then stop. Else go to Step 2.

**Figure 1: Tabu Search pseudo code.**

When selecting the next move to perform, a Tabu Search evaluates the neighborhood of the current solution and attempts to find the best non-tabu move; "best" being determined by the objective value of the resulting solution, should the move be applied. Sometimes, however, it may be desirable to allow a tabu move to be chosen. The conditions under which a tabu move would be allowed are known as the Aspiration Criteria. The most common aspiration criteria is to test whether or not the implementation of the tabu move would result in the best-fit solution yet found for the current run. This is the criteria used by Battiti and Tecchiolli in the RE-TS. Figure 1 shows the basic elements of TS.

During a Tabu Search run, it is possible that a single solution will be visited multiple times. To some degree this is desirable; it supports the concepts of exploitation and exploration. On repeated visits of a solution, the Tabu List will most likely contain a different set of tabu moves, and the search may travel a new path. Problems can arise, however, depending on the length of the Tabu List; a search can get caught in a loop and continuously revisit the same solution. When the chain of moves involved in the loop is longer than the length of the Tabu List, this will result in an infinite loop and the algorithm will spend all of its time evaluating the same solutions repeatedly, leaving large areas of the search space unexplored.

The EE-TS incorporates many of the elements of RE-TS, including a two-level escape mechanism to prevent such infinite loops. This escape mechanism is based on the incorporation of long-term memory within the RE-TS. For each solution that is visited, a corresponding record of the solution is maintained along with a counter. Each time a solution is revisited, the counter is incremented. If a solution has been visited more than some predefined number of times the first level of the escape mechanism passes for that solution. The second level passes when a predefined number of solutions have passed the first level. When the second level passes, an escape occurs: the Tabu List is emptied and a new solution is generated randomly. The revisited solution counters are reset as well. The search effectively restarts but maintains the current iteration count.

## 4. Evolutionary Computing Concepts and Operators in EE-TS

Evolutionary Computing strategies are based on concepts associated with the natural process of evolution [17]. Traits are passed from parents to offspring with operators that mimic selection and recombination. Over time, those traits that are undesirable for survival will be weeded out of a population while "good" traits become prevalent. This "Survival of the Fittest" approach has been successful when applied to many complex problems [18]. Here, we introduce concepts that have been incorporated to some degree into the design of EE-TS. The specific details are discussed in the next section.

Quite often, the search space for a problem will contain multiple local optima. Such a problem is considered to be multimodal. It is possible for an algorithm to become trapped in one of these local optima and be unable to discover the global optimum. Mutation and random restart are techniques used to try to avoid this situation. Another approach is the use of a technique that has been designed for multimodal landscapes, such as the Multi-Niche Crowding Genetic Algorithm (MNC-GA) [19], which naturally encourages the creation of species that converge to multiple niches in the search space. MNC-GA is a genetic algorithm that replaces fitness proportionate reproduction with crowding selection and introduces a replacement technique known as Worst-Among-Most-Similar (WAMS) to encourage competition among members of the same niche.

In crowding selection most individuals in the population get a chance for mating in every generation. Application of this selection rule is done in two steps. First, an individual from the population is selected at random as a parent for mating. Second, its mate is selected, not from the entire population, but from a small group of individuals of size $C_s$ (crowding selection group size), picked uniformly at random (with replacement) from the population. The mate thus chosen must be the one who is most "similar" to the selected individual. The similarity metric used here is not a genotypic metric such as the Hamming distance, but a suitably defined phenotypic distance metric.

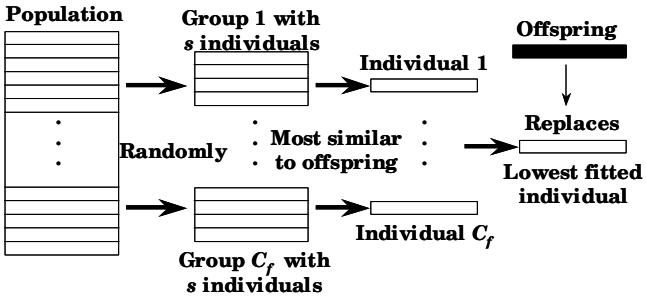


**Figure 2: WAMS replacement strategy.**

The WAMS replacement strategy (Figure 2) is implemented by running a series of tournaments, finding a winner for each tournament, and then having the tournament winners compete against each other for the position of series winner. First, $C_f$ "crowding factor groups" are created by picking uniformly at random (with replacement) $s$ (crowding group size) individuals per group from the population. Second, one individual from each group that is most similar to the offspring is identified. This gives $C_f$ individuals that are candidates for replacement by virtue of their similarity to the offspring. The offspring will replace one of them. From this group of most similar candidates, we pick the one with the lowest fitness to die and be replaced by the offspring. The series winner's is replaced by the newly-generated individual.

The MNC-GA is an algorithm that (a) maintains stable subpopulations within different niches, (b) maintains diversity throughout the search, and (c) converges to different local optima. No prior knowledge of the search space is needed and no restrictions are imposed during selection and replacement thus allowing exploration of other areas of the search space while converging to the best solutions in the different niches.

## 5. The Enhanced Evolutionary Tabu Search

The general structure of EE-TS is based on RE-TS. Unless noted, EE-TS operates in the same way. Consult [1] for a detailed description of RE-TS. Here, we only describe the enhancements implemented in EE-TS.

Figure 3 depicts the overall EE-TS process. Unique elements of the EE-TS include: (1) the generation of an initial population at the beginning of the algorithm and during the escape mechanism,

(2) the running of a tournament series and the processing associated with the result for each iteration, and (3) the running of a tournament series in order to choose a new solution during the escape mechanism. The main goal of the enhancements is to introduce an evolutionary mechanism that contributes to the balance between exploring and exploiting the search space.

The original RE-TS implements a purely random mechanism for generating new solutions. In EE-TS, new solutions are generated using a heuristic operator. The EE-TS heuristic operator assigns facilities to locations by attempting to place facilities with high shipping costs into locations that are on average closer to other locations. Randomness is introduced into the heuristic operator in order to ensure that a variety of solutions are generated. A Restricted Candidate List (RCL) approach is used where some percentage of the best candidate assignments are eligible to be chosen as the next assignment. First, the unassigned location with the lowest average distance to the other locations is chosen. The RCL is then built by selecting the top 25% of unassigned facilities with the lowest total shipping costs. One of the facilities from the RCL is chosen at random and assigned to the location. This process repeats until all locations have been assigned a facility.

```
Step 1: Generate initial Population P.
Step 2: Set i and i* to best solution in P.
Step 3: Set escape := checkForRepetitions( i ).
Step 4: if escape = FALSE then
            evaluateNeighborhood().
            move := chooseBestMove().
            champion := runSeriesOfTournaments().
            child := crossOver(i, champion).
            if fitness(child) < fitness(i, move) then
               i := child.
            else
               i := applyMove(i, move).
Step 5: else
            champion := runSeriesOfTournaments().
            i := crossOver(i, champion).
            resetTabuListAndSolutionHistory();
Step 6: If fitness(i) < fitness(i*) then set i* = i.
Step 7. If a stopping condition is met then stop. Else go to Step 3.
```

**Figure 3: EE-TS Pseudo Code**

To complete the creation process, a greedy local search is performed on the new solution to find a local minimum. This minimum is used as the new solution. Local search usually involves the evaluation of a neighborhood followed by a transition into the most-fit neighbor. Instead of evaluating the entire neighborhood and choosing the best move, the greedy local search used in EE-TS begins evaluating the neighbors, but always accepts the first one found that has a better fitness than the current solution. Each time a better neighbor is found the current solution transitions to the neighbor and the search begins again. The search is complete when an entire neighborhood is evaluated and the current solution has a better fitness than any of its neighbors. Figure 4 provides the pseudo code for the greedy local search technique in EE-TS.

By their nature, Tabu Search tends to have a limited amount of stochastic behavior. Some randomness does exist in the creation of initial solutions and in the size of the Tabu List for a Robust Tabu Search, but compared to many other Soft Computing techniques Tabu Search is a fairly predictable process. Some work has been done on introducing more randomness into Tabu Search [16]. The EE-TS increases the amount of stochastic behavior in the following manner: during each iteration of EE-TS, after a move is selected, a new solution is generated with some randomness. If this new solution has a worse fitness than that which would result from applying the selected move to the current solution, then the move is accepted and the Tabu List is updated in the usual manner defined by the RE-TS. If the new solution is better than the move, however, then the new solution is accepted and nothing is added to the Tabu List. The effect on the Tabu Search is similar to the Evolutionary Computing concept of mutation; a higher degree of randomness is introduced into the normal processing flow. In addition, the new solution contains characteristics of previously-visited good solutions, with the intent that these characteristics contribute to the exploration and exploitation of new, well-fit areas of the search space.

```
Step 1: Set i := 0
Step 2: Set j := i + 1
Step 3: Set moveValue := cost_of_swap(location i, location j).
Step 4: If moveValue > 0 then
            Swap facilities at locations i & j.
            Go to step 1.
Step 5: Set j := j + 1.
Step 6. If j < N-1 then go to step 3.
Step 7: Set i := i + 1.
Step 8. If i < N-1 then go to step 2.
```

**Figure 4: Greedy local search pseudo code.**

The mechanism by which these new stochastic moves are generated in the EE-TS is based on common Evolutionary Computing operations: selection and recombination. Instead of generating a completely random (or greedy) solution for each iteration, the EE-TS recombines the current solution with another randomly-selected, well-fit solution. The resulting offspring fitness is compared with the selected best move from the current solution. Recombination is performed such that common traits are retained in the offspring—if both parents are well-fit it is likely that the common traits are desirable. The resulting offspring solution may be a good solution that is some distance away from the current solution, in a new area of the search space. A greedy local search is then performed on the offspring solution to transition it to a local minimum. If the new solution is better than the current move then it is accepted over the selected move. The algorithm in effect jumps to the new solution and continues searching from there.

Selection is performed in the EE-TS using a tournament technique similar to the WAMS operator described previously. A series of tournaments is run, and the winner of the series is the individual selected for use in the recombination operation along with the current solution. The winner of each discrete tournament is chosen based on similarity, measured in relation to the current solution. The winner of a tournament series is determined based on fitness. Unlike WAMS, which is used in conjunction with the crowding selection technique in order to assist with the fostering and management of multiple niches within a population, the EE-TS technique is implemented with a complementary recombination approach to further enhance the Tabu Search's ability to explore and exploit the search space.

Similarity is based on the number of facilities assigned to the same location. If, for each location, the facility assigned in solution *A* does not match the facility assigned in solution *B*, the similarity score is incremented by one. Two identical solutions will have a similarity of zero.

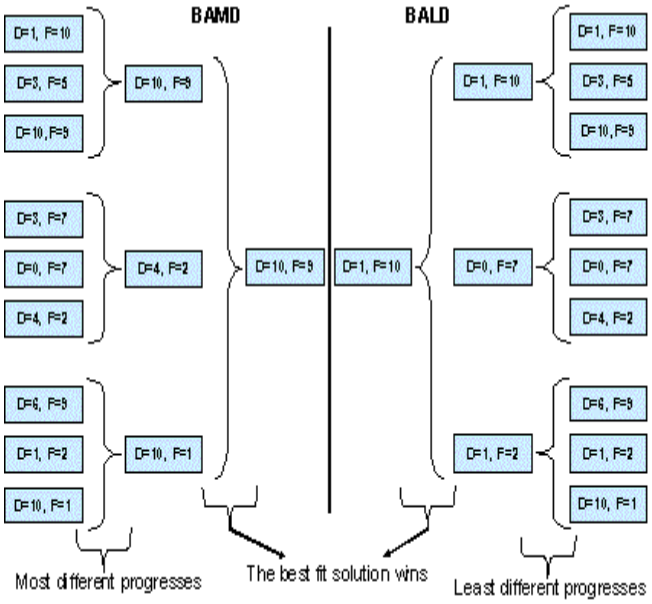


**Figure 5: BAMD and BALD Techniques.**

Two distinct selection approaches were tried during the development of EE-TS and are depicted in Figure 5. They are: Best-Among-Most-Different (BAMD) and Best-Among-Least-Different (BALD). The idea behind BAMD is that by selecting a solution for recombination that has little in common with the current solution, but that still has a good fitness, the algorithm will be lead into new and unexplored areas of the search space. For smaller-sized problems, where it is more likely that individuals will have many facility/location assignments in common, the BAMD technique would seem to be especially useful. It is important to note with this technique, however, that as problem size increases it becomes more likely that the most different individual will have no facility-location assignments in common with the current solution at all. If this should occur, it would negate the propagation of good characteristics to any offspring, since no common traits would exist in the parents. To counter such behavior, the BALD technique was developed for use with larger-sized problems, where it is more likely that the least different individual will have very few (if any) facility-location assignments in common with the current solution. It can be observed that BALD is a niching technique similar to WAMS.

The contestants in the individual tournaments are chosen randomly with replacement, from the current run's population. This population consists of all previously-visited solutions for the current run of the EE-TS algorithm. Unlike the populations of most traditional Evolutionary Computing techniques, the EE-TS population does not maintain a constant size. Instead, it continues to grow during the course of a run. The population size decreases only when an escape occurs, at which time the population is cleared.

Special attention must be paid to the early iterations of the EE-TS: Since the tournaments are carried out with individuals from the population, and the population is made up of previously-visited solutions, the population is not adequately-sized at the beginning of a run in order for a series of tournaments to be worthwhile. In order to hold a meaningful series of tournaments at this early stage, the EE-TS incorporates the idea of an initial population into its design. At the beginning of a run of the algorithm, before any Tabu Searching functionality is performed, a set of random solutions is generated using the greedy RCL technique described previously. The best solution in the initial population is used as the starting point of the run, the others are treated as previously-visited solutions and are available to take part in any of the tournaments held throughout the course of the run. Additionally, whenever an escape occurs and the list of previously-visited solutions is cleared, an initial population is generated before Tabu Searching resumes.

In the interest of minimizing the amount of time spent on the recombination operation, it is implemented with a fairly simple approach: any facility-location assignments that are found in both parents are passed on to the offspring. All remaining unassigned facility-location assignments in the child are chosen at random. The final step in this recombination strategy is a greedy local search on the new offspring.

Including this recombination process at each iteration augments the way in which the Tabu Search explores and exploits the search space. If the two solutions being recombined are very similar, the offspring will be exploiting those characteristics. If the two solutions being recombined have little in common, the offspring will most likely be exploring different areas of the search space. Instead of only being aware of the neighborhood of the current solution, the EE-TS is able to consider transitioning to an out-of-neighborhood solution, but one which is derived from information learned over the course of the run.

## 6. Results

Several of the Enhanced Evolutionary Tabu Search's features can be parameterized at runtime, and the settings for these variables could have a significant impact on the performance and behavior of the algorithm. Battiti and Tecchiolli have done a more than adequate job at describing the variables associated with elements of the Reactive Tabu Search, so this paper will focus exclusively on new parameters introduced with the EE-TS enhancements. All EE-TS runs have been performed with the default RE-TS configuration.

The EE-TS exclusive variables are: Population Size, Tournament size, Tournament Series Size, and Selection Strategy.

A separate group of runs was performed for each of these parameters over a collection of five QAP problems from the Taillard set. Each problem increased in size by an increment of 20, with the smallest problem being of size 20 and the largest of size 100.

Tables 1 through 4 contain the results of the examination of the data collected during the parameter discovery test runs. Ten runs were performed for each variable value and problem combination. To determine which value was to be used in the final EE-TS configuration, comparisons between the average of the best fitness recorded for the ten runs were performed. In Tables 1 through 4, the variable value that performed best for each problem is labeled as Best. In the event that multiple values obtained the same best average fitness, the least processor-intensive value was deemed the Best. In most of the tests, all of the results were very close. This is most likely attributable to two factors: the fitness scores were all large numbers—in the hundred-thousands or millions—so even a small percentage difference could be a substantial gap, and the underlying RE-TS structure was not hindered in any way, so that even a poor EE-TS variable configuration would still find good solutions.

**Table 1: Comparison of population size as a function of the number of location and facilities (*N*).**

| Pop Size | Tai20a | Tai40a | Tai60a | Tai80a | Tai100a |
|---|---|---|---|---|---|
| *N* | + 0 % | + 0.120% | + 0.005% | **Best** | + 0.048% |
| *N* / 2 | + 0% | + 0.170% | **Best** | + 0.005% | + 0.019% |
| *N* / 4 | **Best** | **Best** | + 0.009% | + 0.036% | **Best** |

**Table 2: Comparison of tournament size as a function of population size.**

| Tournament Size | Tai20a | Tai40a | Tai60a | Tai80a | Tai100a |
|---|---|---|---|---|---|
| Pop Size | + 0% | **Best** | + 0.074% | + 0.079% | + 0.142% |
| Pop Size / 2 | + 0% | + 0.042% | **Best** | + 0.080% | + 0.080% |
| Pop Size / 4 | **Best** | + 0.055% | + 0.016% | **Best** | **Best** |

The following values were decided upon for use as the final configuration of the EE-TS: The population size was set to one fourth of the problem size; the size of each tournament was set to one fourth of the population size, with a minimum size of two; each tournament series was made up of five tournaments; and the BAMD strategy was used as the selection strategy. In general, the value which lead to the most Best's was selected for use. With the tournament series size variable, five was chosen over seven in order to decrease execution time. The difficulty in choosing the variable values in this way is that it is unlikely that any of the variables impact the algorithm discretely. Instead, the combined configuration of the variables must be observed to discover the ideal values. Due to time constraints, such a thorough investigation was not performed, but would be beneficial to future development of the EE-TS.

**Table 4: Comparison of discrete values for tournament series size.**

| Series Size | Tai20a | Tai40a | Tai60a | Tai80a | Tai100a |
|---|---|---|---|---|---|
| 1 | + 0.061% | + 0.009% | + 0.072% | **Best** | + 0.132% |
| 3 | + 0.030% | + 0.072% | + 0.106% | + 0.044% | + 0.063% |
| 5 | **Best** | + 0.060% | **Best** | + 0.023% | + 0.043% |
| 7 | + 0.061% | **Best** | + 0.084% | + 0.031% | **Best** |

**Table 5: Comparison of selection strategies.**

| Selection Strategy | Tai20a | Tai40a | Tai60a | Tai80a | Tai100a |
|---|---|---|---|---|---|
| BAMD | **Best** | **Best** | **Best** | **Best** | **Best** |
| BALD | + 0.263% | + 0.667% | + 0.804% | + 0.941% | +0 .999% |

The selection strategy results in Table 4 show that the BAMD approach outperformed the BALD approach in every case, especially in the larger problems where it was originally theorized that BALD would help to improve performance. These results deserve further study, but one explanation for the observed behavior is that with the larger problems even the least distant solutions often had no assignments in common with the current solution and that the BALD strategy therefore had no advantage over BAMD. With the smaller problems, it was assumed that BAMD would outperform BALD.

**Table 3: Comparison of average iterations before convergence to best solution for EE-TS, RE-TS, S-TS, and RO-TS**

| Prob. | Max. Iter. *EE-TS / Others* | EE-TS | | RE-TS | S-TS | RO-TS |
|---|---|---|---|---|---|---|
| | | Avg. Iterations | Results | | | |
| Tai9a | 100K / 100K | 102.5 (34.2) | 30/30 | 67.5 (12.0) | 56.2 (8.1) | 31.7 |
| Tai10a | 100K / 100K | 96.2 (15.1) | 30/30 | 256.7 (34.0) | 161.3 (20.7) | 137.1 |
| Tai12a | 100K / 100K | 114.5 (21.7) | 30/30 | 282.3 (51.4) | 477.0 (95.7) | 210.7 |
| Tai15a | 100K / 100K | 1084.8 (205.4) | 30/30 | 1780.3 (319.0) | 3642.2 (308.2) | 2168.0 |
| Tai17a | 100K / 100K | 1196.2 (218.0) | 30/30 | 4133.9 (646.8) | 7364.2 (817.4) | 5020.4 |
| Tai20a | 100K / 500K | 14970.8 (2502.5) | 30/30 | 37593.2 (6012.5) | 25092.9 (6572.2) | 34279 |
| Tai25a | 100K / 1M | 19322.3 (2922.7) | 30/30 | 38989.7 (6236.1) | 20483.9 (3575.0) | 80280.4 |
| Tai30a | 100K / 2M | 31905.3 (5001.4) | 24/30 | 68178.2 (11370.3) | 48919.2 (9055.6) | 146315.7 |
| Tai35a | 100K / 4M | 48457.6 (5690.9) | 10/30 | 281334.0 (48543.5) | 146276.2 (47419.7) | 448514.5 |

A series of runs was performed with the EE-TS based on the published results of various other metaheuristic search techniques. Table 5 shows comparisons between the EE-TS and several other successful Tabu Searches over a set of problems from the Taillard set ranging in sizes from 9 to 30. Thirty runs were performed on each problem for each approach. The average number of iterations required to find the best solution is shown along with the standard deviations in parentheses. The RE-TS, S-TS, and RO-TS data contained in this table was gathered from Battiti and Tecchiolli [1]. For all but the first problem, the EE-TS required fewer iterations to find the optimal solution. For all runs of the EE-TS a maximum of 100,000 iterations was allowed. For the four largest problems, this number differed from that of the other Tabu Search techniques, which were set to run for a much higher number of iterations. For the two largest problems, the EE-TS failed to find the best known solution for some of the 30 runs performed on each problem. Allowing for a greater number of iterations may have improved this area of performance.

In all cases, the runtime environment for the RE-TS, S-TS, and RO-TS tests was either not known or could not be duplicated, so time comparisons are not possible. Still, it should be noted that the EE-TS adds additional processing to the RE-TS, including local searches for each iteration. This undoubtedly has some impact on execution time, and deserves further investigation. Also, no time was permitted for optimization purposes and potential improvements in this area should be reviewed as well.

**Table 6: EE-TS statistical averages.**

| Prob. | Results | Evol.Solution Accepts | Move Accepts | Escapes |
|---|---|---|---|---|
| Nug15 | 30/30 | 29.7 (5.0) | 65.7 (9.7) | 1.0 (0.2) |
| Nug30 | 30/30 | 171.0 (35.4) | 1846.2 (461.3) | 8.5 (1.6) |
| | | | | |
| Sko49 | 23/30 | 487.6 (63.7) | 50354.4 (6534.2) | 60.0 (7.6) |
| Sko56 | 30/30 | 213.0 (37.8) | 21008.1 (3761.8) | 24.8 (4.3) |
| Sko64 | 29/30 | 206.1 (40.5) | 22645.3 (4404.8) | 27.6 (5.1) |
| | | | | |
| Tai9a | 30/30 | 86.0 (33.1) | 15.7 (3.1) | 0.8 (0.2) |
| Tai10a | 30/30 | 54.0 (12.2) | 41.1 (5.7) | 1.1 (0.2) |
| Tai12a | 30/30 | 66.3 (14.9) | 46.6 (7.7) | 1.7 (0.3) |
| Tai15a | 30/30 | 214.3 (37.2) | 864.0 (167.9) | 6.5 (1.1) |
| Tai17a | 30/30 | 220.0 (36.6) | 969.0 (181.9) | 7.3 (1.2) |
| Tai20a | 30/30 | 1563.0 (264.9) | 13359.3 (2230.7) | 48.4 (8.2) |
| Tai25a | 30/30 | 1217.2 (187.9) | 18069.3 (2730.2) | 35.8 (5.7) |
| Tai30a | 24/30 | 2667.0 (376.3) | 43335.4 (6380.0) | 103.7 (14.6) |
| Tai35a | 10/30 | 1753.5 (133.3) | 78251.5 (5891.0) | 88.3 (6.8) |

In addition to the Taillard set problems listed in Table 5, several problems from the Nugent, Vollman, and Ruml and Skorin-Kapov sets were used to conduct runs as well. Table 6 shows some of the statistics collected for tests consisting of 30 runs per problem, including the number of successful runs, the average number of evolutionary solutions accepted during each run, the average number of normal moves accepted for each run, and the average number of escapes performed for each run. Numbers in parentheses are standard deviations. Of the fourteen problems, only four tests failed to find the best known solution for all 30 runs. Interestingly, all runs for the Nug30 problem discovered the best known result, a feat not duplicated by any of the approaches examined in Merz and Freisleben's comparison of various metaheuristic search techniques [20]. Also interesting to note is that for problems with sizes less than 15, the number of evolutionary solutions accepted was greater than the number of normal moves accepted. With these smaller problems, it is likely that the recombination technique—which included the greedy local search—was very successful at quickly finding very good minima, much more so than the un-enhanced Tabu Search elements of the EE-TS.

A final observation on Table 6 is that the average number of escapes is fairly small for each of the problems. This implies that few solutions were visited multiple times. Also, for the larger problems the average number of normal move acceptances is far greater than the number of evolutionary move acceptances, which indicates that the pure Tabu Search features of the EE-TS had more success at identifying better-fit solutions within the search space. One of the strengths of the EE-TS design is that it contains more than one mechanism by which to manage the exploration and exploitation of the search space—one inherited from the RE-TS's dynamic Tabu List length and escape mechanism and one gained from the generation of new solutions via recombination—and that these mechanisms have the potential to complement or compensate for each other over the course of any given run. The recombination operation gives the algorithm the opportunity to consider solutions that may be outside of the neighborhood of the current iteration, but which demonstrate good characteristics based on what has previously been seen over the course of a run. When the recombination technique is repeatedly successful at discovering good solutions, its results can dominate the flow of the algorithm. When it is not, the Tabu Search will continue to explore and exploit the search space in its own way.

**Table 7: Results for larger problem sizes with number of iterations of 100K.**

| Prob. | Results | Avg. Best | Distance from Known Best | Avg. Iterations |
|---|---|---|---|---|
| Tai40a | 0/30 | 3154048.1 (670.3) | + 0.4675% | 53528.7 (5200.5) |
| Tai60a | 0/10 | 7272281.4 (2737.0) | + 0.9203% | 69505.6 (8468.8) |
| Tai80a | 0/10 | 13637294.8 (4203.7) | + 0.7486% | 50067.4 (6042.1) |
| Tai100a | 0/10 | 21263584.4 (6303.6) | + 0.7486% | 62542.5 (8754.1) |

Finally, a series of runs was performed on a collection of larger problems. Due to the amount of processing time required to run these tests, only 10 runs per problem were performed in most cases. In all cases, the maximum number of iterations allowed was 100,000. Generally, a much greater number of iterations (sometimes millions) is needed by other established metaheuristic search techniques to discover the best known solutions. Not surprisingly, none of the tests described in Table 7 found the best known solution to the respective problems. What is shown, however, is that for all of the runs the average best found solutions came within less than 1% of the best known solutions.

The best known solution fitnesses were taken from the QAPLIB [5].

## 7. Conclusions and Recommendations

This paper has described a new metaheuristic technique based on elements of Tabu Searching and Evolutionary Computing. The Quadratic Assignment problem was used during development of the new algorithm for tuning purposes, and also to compare its performance with established searching techniques. The Enhanced Evolutionary Tabu Search (EE-TS) augments the exploration and exploitation of a search space inherent in the RE-TS by introducing selection and recombination operations that use information about good solutions discovered during a run of the algorithm to influence the search. Through demonstration, it was shown that the Enhanced Evolutionary Tabu Search was able to reach an optimum value—the best known value, in most cases—of a problem in fewer iterations than other well-known techniques. While no solutions have yet been found with the EE-TS that are better than any previously known solutions, the fact that fewer iterations were required for successful searches implies that in some scenarios, the EE-TS may be a better choice than the established techniques.

Based on these results, further investigation is deserved of the EE-TS algorithm. Specifically, a more thorough examination of the impact of the configuration variables and their relationships could lead to better performance. Also, promising initial tests require longer runs (more iterations) be attempted on larger problems to observe comparable performances with other approaches. Additional areas that would benefit from more research include the effects of the BALD technique on larger problem sizes, the impact of a randomly-generated escape solution, the evaluation and possible improvement of execution time, and the development of optimization strategies for the algorithm's source code.

## 8. ACKNOWLEDGMENTS

The authors thank to Lockheed Martin Integrated Systems & Solutions and the Molecular Design and Informatics group at Johnson & Johnson Pharmaceutical R&D for providing us with the time and support to carry out this research. The authors thank the anonymous reviewers for their many valuable comments and for bringing to our attention some relevant published work.

## 9. REFERENCES

[1] Battiti, R. and Tecchiolli, G. The reactive tabu search. *ORSA Journal on Computing, 6, 2*, 126-140, 1994.

[2] Glover, F. and Kochenberger, G. A. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003.

[3] Nugent, C.E., Vollmann, and T.E., Ruml, J. An experimental comparison of techniques for the assignment of facilities to locations. *Operations Research*, 16:150-173, 1968.

[4] Resende, M., Pitsoulis, L., Pardalos, P. Fortran subroutines for computing approximate solutions of weighted MAX-SAT problems using GRASP. *Discrete Applied Mathematics,* 100: 95-113, 2000.

[5] Burkhard, R.E., Karish, S.E., and Rendl, F. QAPLIB – A quadratic assignment problem library. *Journal of Global Optimization*, 10:391-403, 1997.

[6] Sahni, S. and Gonzalez, T. P-complete approximation problems. *J. ACM 23*, 555-565, 1976.

[7] Anstreicher, K.M. Recent advances in the solution of quadratic assignment problems. *Mathematical Programming, Series B* 97:27-42, 2003.

[8] Taillard, E. D. Robust tabu search for the quadratic assingnment problem. *Parallel Computing*, 17:443-455, 1991.

[9] Skorin-Kapov. Tabu search applied to the quadratic assingnment problem. *ORSA Journal on Computing*, 2(1):33-45, 1990.

[10] Taillard, E. Comparison of iterative searches for the quadratic assignment problem. *Location Science*, 3:87-103, 1995.

[11] Glover, F. Future Paths for Integer Programming and Link to Artificial Intelligence. *Computers and Operations Research*, 12:533-549, 1986.

[12] Glover, F. Tabu search – part I. *ORSA Journal on Computing*, 1(3): 109-206, 1989.

[13] Glover, F. Tabu Search – part II. *ORSA Journal on Computing*, 2:4-32, 1990.

[14] Glover, F., Laguna, M. *Tabu Search*. Kluwer Academic Publishers, 1997.

[15] Glover, F. Laguna, M. Tabu Search. *Modern Heuristic Techniques for Combinatorial Problems*. Colin Reeves, ed. Blackwell Scientific Publishing, 71-140, 1993.

[16] De Falco, I., Del Balio, R., Tarantino, E., and Vaccaro, R. Improving search by incorporating evolution principles in parallel tabu search. *IEEE World Congress on Computational Intelligence,* 2:823-828, June 27-29, 1994.

[17] Holland, J.H. *Adaptation in Natural and Artificial Systems*, 2nd Edition, Cambridge: MIT Press, 1992.

[18] Mitchell, M. *An Introduction to Genetic Algorithms*, Cambridge: MIT Press, 1996.

[19] Cedeño, W. *The Multi-Niche Crowding Genetic Algorithm: Analysis and Applications*, Ph.D. Dissertation, Computer Science Department, University of California, Davis, September 1995. UMI Dissertation Services, Microfilm Number 9617947.

[20] Merz, P. and Freisleben, B. A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. *In Proceedings of the 1999 International Congress of Evolutionary Computation* (CEC'99), Washington DC, USA, 1999.