Using Data Mining Technique for Coefficient Tuning of an Adaptive Tabu Search

Elena Ikonomovska[*], Dejan Gjorgjevik[*], Suzana Loskovska[*]
[*] University Ss. Cyril & Methodius, Faculty of Electrical Engineering and Information Technologies, Skopje, Republic of Macedonia, elenai@feit.ukim.edu.mk, dejan@feit.ukim.edu.mk, suze@feit.ukim.edu.mk

Abstract— **This paper describes the Adaptive Tabu Search algorithm (A-TS), an improved tabu search algorithm for combinatorial optimization. A-TS uses a novel approach for evaluation of the moves, incorporated in a new complex evaluation function. A new decision making mechanism triggers the evaluation function providing means for avoiding possible infinite loops. The new evaluation function implements effective diversification strategy that prevents the search from stagnation. It also incorporates two adaptive coefficients that control the influence of the aspiration criteria and the long-term memory, respectively. The adaptive nature of A-TS is based on these two adaptive coefficients. This article also presents a new data mining approach towards improving the performance of A-TS by tuning these coefficients. A-TS performance is applied to the Quadratic Assignment Problem. Published results from other authors are used for comparison. The experimental results show that A-TS performs favorably against other established techniques.**

Keywords—**data mining, heuristic, coefficients tuning, tabu search, quadratic assignment problem.**

I. INTRODUCTION

Heuristic search algorithms have proven to be very useful in solving difficult combinatorial optimization problems. Due to their ability to escape local optima, most successful heuristic local search techniques are Simulated Annealing, Genetic Algorithms, and Tabu Search with its variations. Tabu Search has been very successful in achieving near-optimal (and sometimes optimal) solutions to a variety of hard problems.

This paper introduces the Adaptive Tabu Search (A-TS), an improved tabu search algorithm for combinatorial optimization. Adaptive Tabu Search introduces a new evaluation function to the basic scheme of Tabu Search. Our Tabu scheme also proposes a new mechanism for selecting the best move. The selection process uses the evaluation function, which incorporates both long-term memory and aspiration criteria. The evaluation function used in this process involves several parameters and coefficients. Choosing appropriate values for these parameters has great impact on A-TS performance. To achieve better performance, coefficient tuning was performed applying data mining techniques.

The performance of our A-TS is evaluated by using instances of the Quadratic Assignment Problem (QAP), chosen from the QAP Library (QAPLIB) [1]. By solving the same problem instances of QAP used by other cited researchers [2][3][4][5], we aimed to derive objective conclusions of the advantages of our Adaptive Tabu Search and of the use of data mining techniques for optimizing the performance of meta-heuristic algorithms.

Section II presents a formal definition of the QAP. Section III provides a brief overview of the basic Tabu Search algorithm and its popular variations. In section IV, we describe the main improvements that we propose to the basic TS algorithm, resulting in our Adaptive Tabu Search (A-TS) algorithm. The environment used to test A-TS is described in section V. The implementation issues of a data mining technique for coefficients tuning are addressed in section VI. Section VII presents the experimental results. Our conclusions and areas of further research are given in section VIII.

II. THE QUADRATIC ASSIGNMENT PROBLEM

The Quadratic Assignment Problem (QAP) is NP-hard combinatorial optimization problem [6]. Its many practical instances come from areas such as design and resource allocation, microprocessor design and scheduling. Due to the complexity of QAP, in some ways, QAP has become a benchmark by which new techniques are validated.

For the first time, Koopmans and Beckman stated QAP in 1957 [7]. It can be described as follows: Given two $n\times n$ matrices $A=(a_{ij})$ and $B=(b_{ij})$, find a permutation π^* minimizing

$$\min_{\pi\in\Pi(n)} f(\pi)=\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\cdot b_{\pi(i)\pi(j)}$$

where $\Pi(n)$ is the set of permutations of n elements.

In other words, it deals with identifying optimal assignments of facilities to locations such that the cost of the resulting system is minimized. Shani and Gonzalez [6] have shown that the problem is NP-hard and that there is no ε-approximation algorithm for the QAP unless P = NP.

While some NP-hard combinatorial optimization problems can be solved exactly for relatively large instances, QAP instances of size larger than 20 are considered intractable. In practice, a large number of real world problems lead to QAP instances of considerable size that cannot be solved exactly. For example, an application in image processing requires solving more than 100 QAP problems of size $n = 256$ [8]. Even with today's fastest computers, relatively small problems require prohibitive amounts of time to solve to provable optimality [9]. The use of heuristic methods for solving large QAP instances is currently the only practicable solution.

III. Tabu Search Overview

Glover introduced Tabu Search (TS) in the late 80's [10][11][12]. The basic idea behind TS is that, adding short-term memory to 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) [5], which introduced a dynamic randomly sized short-term memory design. Battiti and Tecchiolli developed the RE-TS [2] in 1994. They introduced a dynamically sized short-term memory, dependent on the runtime characteristics of the algorithm. In addition, they utilized a form of long-term memory that helped prevent searches from stagnating.

Many other TS variations have been developed that incorporate various forms of dynamically sized short-term memory and long-term memory [13][14]. Still, the RO-TS and RE-TS remain among the most successful and popular. The following concepts are common to most (if not all) Tabu Search techniques, but their specific implementations are somewhat flexible.

A move *m* is an operation by which one solution is transformed into a new, neighboring solution. The neighborhood of the solution, $N(i,k)$, is the set of all solutions that can be derived 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 implements the short-term memory. It is the most influential piece of any TS design. The basic purpose of the list is to maintain a record of moves that are tabu (discouraged) during a number of following iterations. 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.

The main weakness of TS is its tendency to explore a too limited region of the search space, i.e., the search lacks breadth, unless systematic and effective diversification schemes are used [15]. During a TS run, it is possible that a single solution will be visited multiple times. To some degree, this behavior 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. However, the problem arises when the algorithm continuously revisits the same set of solutions repeatedly (infinite loop), leaving large areas of the search space unexplored. By increasing the length of the list, the probability of entering an infinite loop decreases. On the other hand, longer lists limit the exploration of the search space. The so-called long-term memory has a great deal in solving this problem.

When selecting the next move to perform, TS evaluates the neighborhood of the current solution and attempts to find the best non-tabu move; "best" being determined as 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 a criterion is to test whether the implementation of the tabu move would result in the best-fit solution yet found, for the current run. Battiti and Tecchiolli used the above criterion in the RE-TS. Fig.1 shows the pseudo code of TS:

```
TABUSEARCH()
  1 Create an initial solution $i$ at random. Set $i^*=i$ and $k=0$.
  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 ($r=1,...,a$).
  3 Choose a best $j=i\in m$ in $V^*$ (with respect to objective function f)
    and set $i=j$.
  4 If $f(i) < f(i^*)$ then set $i^*=i$.
  5 Update tabu and aspiration conditions.
  6 If a stopping condition is met then stop. Else, go to step 2.
```

Fig. 1. The pseudo-code of Tabu Search algorithm.

IV. The Adaptive Tabu Search

The Adaptive Tabu Search, that we propose, explores the meaning of finding the "best" move. The search for the best move is a very computation demanding operation. Therefore, it plays a major part in the speed and accuracy of the solving process. The local search in TS consists of evaluating all moves applicable to the current solution, and choosing the best one. In the A-TS approach, the non-tabu move that generates the greatest improvement of the objective function is chosen and applied. In this case, no aspiration criteria are being utilized. However, in some instances, none of the evaluated non-tabu moves provides any improvement. The proposed evaluation function is triggered only when all evaluated moves are tabu or non-improving, non-tabu. The move for which the evaluation function returns the lowest value is accepted and performed.

Any implementation of TS must provide a balance between exploring and exploiting the search space. The risk of visiting certain solutions infinite number of times must be avoided. On the other hand, the potential benefit from revisiting a single solution has to be encouraged. The aim of A-TS is to achieve this balance and maintain it throughout the whole search.

The evaluation function makes its decisions considering the long-term memory and the remaining time for the move as tabu (*tabu_time_left*). The long-term memory is implemented as a list of counters, remembering the application of each possible move during the search. In the evaluation function, the number of occurrences of the move (*frequency*) is multiplied with an adaptive coefficient (k_1). The value of k_1 is proportional to the value of the move itself (*move_value*), the frequency of the application of the move and the current iteration. The main objective of this adaptive coefficient is to prevent the search from being caught in an infinite loop. This is done by discouraging moves that have been frequently applied. This implements successful diversification strategy, as it will be shown in section 7.

On the other hand, the function includes an aspiration criterion. It allows a tabu move to be performed, if it seems promising and not risky in terms of loops or local stagnation. This criterion is implemented using another adaptive coefficient (k_2), whose value also changes and is proportional to the value of the move. This is because a tabu move with value much greater than the rest of the nonimproving, nontabu or tabu moves in a current iteration will be the best move according to the evaluation function even if it has been applied very recently. When applied the adaptive coefficient k_2, a move will be discouraged according to the value of the move.

The adaptive nature of our Tabu Search scheme is based on these two adaptive coefficients. Their values change with every iteration. The final form of the evaluation function is:

$$evaluation\_function(move\_value, frequency, tabu\_time\_left) = move\_value + k_1 \cdot frequency + k_2 \cdot tabu\_time\_left \quad (1)$$

where:

$$k_1 = \begin{cases} c_1 \cdot iter \cdot \max(move\_value, 1), & \text{if } freq > avgfreq \\ c_2 \cdot abs(move\_value), & \text{if } freq \le avgfreq \end{cases} \quad (2)$$

$$k_2 = \begin{cases} 0, & \text{if } \textit{move is not tabu} \\ c_2 \cdot abs(move\_value), & \text{if } \textit{move is tabu} \end{cases} \quad (3)$$

and *iter* is the current iteration.

The coefficients k_1 and k_2 control the influence of the move frequency and the remaining time of the move in the tabu list. These coefficients drive the heuristic. Therefore, their influence upon the accuracy of the obtained solutions is significant. Since their values also depend on the values of coefficients c_1 and c_2, scientific method should be applied for fine-tuning these coefficients.

V. Benchmark Instances

The problem instances used in the development and testing of A-TS are obtained from the QAPLIB, a public library of QAP problems and their best-known solutions [1]. The problems are organized into sets, with each set named after the author(s) who developed the group of problems. The number in the problem's name corresponds to the size of the problem. QAPLIB currently contains over 100 instances that have been used in earlier researches. Some of them originate from real life applications, like hospital layout (kra30*, els19), typewriter design (bur26*), etc. Most of these problems come from practical applications or they are randomly generated with non-uniform laws that imitate the distributions observed on real world problems.

As shown by Taillard [8], the quality of solutions produced by heuristic methods strongly depends on the problem type, that is, the structure of the data matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. For problems taken from the real world, many heuristic methods perform rather poorly. They are not able to find solutions within 10% of the value of the best solutions known, even if excessive computing time is allowed. Moreover, the poor performance occurs even for small size problems. Conversely, the same methods may perform very well on randomly generated problems. For such problems, almost all heuristic methods are able to find high quality solutions (i.e., solutions approximately one percent worse than the best solution known). Therefore, it is reasonable to analyze the performance of A-TS by splitting the problem instances into two categories: (i) real world, irregular and structured problems, and (ii) randomly generated, regular and unstructured problems.

VI. Coefficients Tunning

The performance of tabu search algorithms depends of many search parameters such as tabu tenures, move selection probabilities, coefficients etc. Significant performance benefits can be achieved by determining appropriate values for these search parameters. In this paper, we present an implementation of a data mining method for optimizing the coefficients c_1 and c_2 incorporated in the evaluation function, to improve the performance of A-TS.

When improving the performance of a certain algorithm the goal is to increase the accuracy and minimize the time needed for producing satisfactory solutions. The coefficients c_1 and c_2 are part of the evaluation function that plays a major role in the speed and accuracy of the search. Determining the relationships between the coefficients c_1 and c_2 and the number of iterations needed for producing optimal or near optimal solutions, would provide valuable information for improving the performance of the algorithm. This kind of relationships or associations can be discovered by the use of data mining algorithms for mining association rules in large data sets. The process is known as association rule mining.

The process of inducing association rules consists of two steps:

Step1: *Finding all frequent itemsets.*
Step2: *Generating strong association rules from the frequent itemsets.*

Among the best algorithms for mining Boolean association rules in large sets of data is the Apriori algorithm [16]. The algorithm uses prior knowledge of frequent itemset properties to prune (reduce) the search space. The name of the algorithm is based on this property.

An *item* is a triple that represents either a categorical attribute with its value, or a quantitative attribute with its range. The value of a quantitative attribute can be represented as a range where the upper and lower limits are the same. We use the term *itemset* to represent a set of items. A *k-itemset* is an itemset that contains *k* items. An itemset is *frequent* if it satisfies minimum support (*min_sup*) threshold, which can be set by user or domain expert. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items. Let *D*, the task relevant data, be a set of transactions where each transaction *T* is a set of items such that $T \subseteq I$. An itemset satisfies minimum support if the occurrence frequency of the itemset is greater than or equal to the product of min_sup and the total number of transactions in *D*.

The Apriori algorithm uses an important property called the *Apriori property* that states: *All non-empty subsets of a frequent itemset must also be frequent.* The property is based on the following observation. By definition, if an itemset I does not satisfy the minimum support threshold, *min_sup*, then I is not frequent, If an item A is added to the itemset I, then the resulting itemset {i.e., $I \cup A$} cannot occur more frequently than I. Therefore, $I \cup A$ is not frequent either. This property is used to improve the efficiency of the level-wise generation of frequent itemsets. A level-wise generation is an iterative search, where k-itemsets are used to explore ($k+1$)-itemsets. First, the set of frequent 1-itemsets is found. This set is denoted L_1. L_1 is used to find L_2, the frequent 2-itemsets, which is used to find L_3, and so on, until no more frequent k-itemsets can be found.

The generation of frequent itemsets is a two-step process, consisting of join and prune actions.

In the join step, to find a frequent set of k items L_k, a set of candidate k-itemsets is generated by joining L_{k-1} with itself. This set of candidates is denoted C_k. The join is performed where members of L_{k-1} are joinable, that is, if they have (k-2) items in common.

In the prune step, the Apriori property is employed to remove candidates that have a subset that is not frequent. The candidate set C_k is a superset of L_k. Its members may or may not be frequent, but all of the frequent k-itemsets are included in C_k. A scan of the database to determine the count of each candidate in C_k would result in the determination of L_k (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to L_k). C_k, however, can be huge, and so this could involve heavy computation. To reduce the size of C_k, the Apriori property is used as follows. Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any (k-1)-subset of a candidate k-itemset is not in L_{k-1}, then the candidate cannot be frequent either and so it can be removed from C_k.

The pseudo-code for the Apriori algorithm is given on Fig. 2.

Once the frequent itemsets from the transactions in D have been found, it is straightforward to generate strong association rules from them. To define a strong association rule, additional definitions must be introduced.

Let A be a set of items. A transaction T is said to contain A if and only if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$ and $A \cap B = \varnothing$. The rule $A \Rightarrow B$ holds in the transaction set D with support s, where s is the percentage of transactions in D that contain $A \cup B$. The rule $A \Rightarrow B$ has confidence c in the transaction set D if c is the percentage of transactions in D containing A which also contain B. That is,

$$Support\ (A \Rightarrow B) = prob\ \{A \cup B\} \tag{4}$$

$$Confidence\ \ (A \Rightarrow B) = prob\ \{B \mid A\} \tag{5}$$

Rules that satisfy both a minimum support threshold (*min_sup*) and a minimum confidence threshold (*min_conf*) are called strong.

In this specific problem, the transaction set D was generated by performing 10000 runs of the algorithm A-TS. Ten different values for coefficients c_1 and c_2 were used and one hundred different initial solutions as a starting point for the search. The values for coefficient c_1 vary in the interval [10, 100]. This is an arithmetic progression where the first term is 10 and the common difference is 10. The values for coefficient c_2 are in the interval [1000^{-1}, 100^{-1}]. Here the base changes with arithmetic progression where the first term is 1000 and the common difference is -100. One hundred different seed values for generating the initial solution were used. The table obtained consists of five quantitative attributes (the number of iterations named *num_iterations*, the difference between the produced and the optimal solution named *gap*, *seed*, c_1 and c_2).

As it was previously stated, the Apriori algorithm is an influential algorithm for mining frequent itemsets for Boolean association rules. A boolean association rule concerns associations between the presence or absence of items. A quantitative association rule describes associations between quantitative items or attributes. Since our data set consists of quantitative attributes, we refer to this mining problem as Quantitative association rules problem.

```
APRIORI($D$: database of transactions, $min\_sup$: minimum support)
 1 $L_1$ is the set of frequent 1-itemsets;
 2 k = 2;
 3 while $L_{k-1}$ is not empty do{
 4     $C_k$ is the candidate set, generated by joining $L_{k-1}$ with itself;
 5     for each transaction t $\in$ $D$ { // scan $D$ for counts
 6     $C_t$ = subset($C_k$ , $t$); // get the subsets of t that are candidates
 7     for each candidate c in $C_t$ test whether it is frequent
 8             c.count++; // scan $D$ in order to determine the
               //count of candidate c
 9     }
 10    $L_k$ = {$c \in C_k$ | c.count $\geq$ $min\_sup$} // $L_k$ is the collection of
       //frequent sets c from $C_k$ that are having count no less than
       //$min\_sup$
 11    k++;
 12 }
 13 return $L = \cup_k L_k$;
```

Fig. 2. The Apriori algorithm for discovering frequent itemsets for mining Boolean association rules.

The Boolean association rules problem can be considered as a special case of Quantitative association rules problem. Therefore, mapping the Quantitative association rules problem into the Boolean association rules problem will allow us to use any algorithm for finding Boolean association rules in order to find Quantitative association rules. If the quantitative attributes in the table have only few values, this mapping is straightforward. Conceptually, instead of having just one field in the table for each attribute, there can be as many fields as the number of attribute values. The value of a boolean field corresponding to <*attribute1, value1*>

would be “1” if attribute1 had value1 in the original record, and “0” otherwise. If the domain of values for a quantitative approach is large, an obvious approach will be to first partition the values into intervals and then map each <*attribute, interval*> pair to a boolean attribute.

There are two problems with this simple approach when applied to quantitative attributes [17]:

1. If the number of intervals for a quantitative attribute (or values, if the attribute is not partitioned) is large, the support for any single interval can be low. Without using larger intervals, some rules involving this attribute may not be found because they lack minimum support.

2. There is some information lost when partitioning values into intervals. Some rules may have minimum confidence only when an item in the antecedent consists of a single value (or a small interval). This information loss increases as the interval sizes become larger.

To solve these problems, we considered all possible continuous ranges over the values of the quantitative attributes. When combining adjacent intervals the first problem disappears. The second problem can be solved by avoiding partitioning of the attributes that are expected in the antecedent of the rule.

For this specific problem, rules will only be interesting if they represent non-trivial correlations between the coefficients c_1 and c_2 as antecedents and the number of iterations and the difference between the produced and the optimal solution as consequents.

To avoid the problem of information loss the attributes c_1 and c_2 are not partitioned into intervals. Instead, they are directly mapped to consecutive intervals, such that the order of the intervals is preserved. Partitioning was done over the attributes *num_iterations* and *gap*, where the number of intervals was kept small.

Three common partition strategies are:

1. equi-width partitioning, where the interval size of each partition is the same,
2. equi-depth partitioning, where each partition has approximately the same number of tuples assigned to it, and
3. homogeneity-base partitioning, where the partition size is determined by the uniform distribution of the tuples in each partition.

Best results were achieved when implementing the equi-width strategy due to the type and structure of the data.

Given a set of records D, the problem of mining quantitative association rules is to find all quantitative association rules that have support and confidence greater than the user-specified minimum support (*min_sup*) and the minimum confidence (*min_conf*) respectively.

We solved the problem of discovering quantitative association rules in five steps:

1. Determining the number of partitions for each quantitative attribute.
2. Partitioning the attributes *num_iterations* and *gap* in the preferred number of intervals respectively.
3. Mapping the values of the quantitative attributes that are not partitioned to consecutive integers such that the order of the values is preserved.
4. Employing the Apriori algorithm for finding all sets of items whose support is greater then the user-specified minimum support. These are the frequent itemsets. These frequent itemset are then used to generate association rules.
5. Determining the interesting rules from the set of previously generated association rules.

To find the most adequate number of intervals for attributes *num_iterations* and *gap,* all the possible variations were considered. From the obtained results, it was concluded that the number of partitions depends of the type and the size of the problem. Precisely, the number of partitions is inversely proportional to the size of the problem. Because the computational time was limited, the accent was given on improving the accuracy of A-TS. Therefore, the number of partitions for the attribute *num_iterations* was kept small. For the attribute *gap,* the number of partitions was set to a value that corresponded to the most interesting rules.

After partitioning the attributes, the Apriori algorithm was employed to find the frequent itemsets and generate the association rules. The value of the parameter min_sup was set to 0.01.

For determining the interesting strong rules from the generated set of association rules, additional measure of interestigness should be applied [19]. The most popular objective measure of interestigness is *lift*. Lift is defined as the ratio of the frequency of the consequent (B) in the transactions that contain the antecedent (A) over the frequency of the consequent in the data as a whole.

$$Lift(A \Rightarrow B) = Confidence(A \Rightarrow B) / Support(B) \quad (6)$$

Lift values greater than 1 indicate that the consequent is more frequent in transactions containing the antecedent than in transactions that do not. It indicates the influence of the antecedent over the frequency of the consequent.

Using this measures, we define an interesting rule as a rule that satisfies the user-specified minimum confidence threshold and has *lift* greater then one. The minimum confidence threshold in this process was set to 0.5.
This process was performed for a subset of QAP problem instances of sizes between n=20 and n=35. For each of them, the interesting rules were extracted and the most promising values for the coefficients c_1 and c_2 were set. To compare the results obtained before and after the coefficients tuning one hundred trials were performed for each problem. Trials were performed only for short runs. The experiments evaluate the improvement of the quality of produced solutions under strong time constraints.

Table I provides a comparison of the quality of solutions before and after tuning. The last column represents the improvement as a difference between the solution quality before and after tuning. Before coefficients tuning, trials were performed for c_1=10 and c_2=100^{-1}. For half of the problem instances the improvement is zero because the optimal coefficients values were the same before and after. For the other half of the cases, the average value of the improvement is 12.954 percent. It was concluded that the value of coefficient c_1 has greater influence on the quality of the solution. This is due to the fact that, k_1 which contains c_1 introduces the long-term memory in the evaluation function of A-TS. The column that represents the difference in iterations shows that in almost all of the cases the improvement is achieved when the number of iterations is increased. This means that, the quality of the algorithm is a compromise between the speed and the accuracy. When coefficient tuning was applied to improve the speed of the algorithm, the number of iterations was decreased, but this induced lower accuracy of the algorithm.

TABLE I.
QUALITY OF A-TS FOR REGULAR AND IRREGULAR QAP PROBLEMS MEASURED IN PERCENT ABOVE THE BEST SOLUTION VALUE KNOWN.

Proble m name	**Best known value**	**Before tuning**	**After tuning**	**Differenc e in iteration s**	Improve ment	In percent
Tai20b	122455319	13.983	13.886	285	0.097	0.69
Tai25b	344355646	3.909	2.589	-1629	1.320	33.768
Tai30b	637117113	4.119	3.707	-468	0.412	10.002
Tai35b	283315445	2.635	2.635	0	0	0
Kra30a	88900	0.583	0.495	-665	0.088	15.094
Kra30b	91420	0.002	0.002	0	0	0
Chr25a	3796	1.452	1.452	0	0	0
Nug20	2570	0	0	0	0	0
Nug30	6124	0.020	0.020	0	0	0
Tai20a	703482	0.250	0.250	0	0	0
Tai25a	1167256	0.814	0.814	0	0	0
Tai30a	1818146	0.371	0.327	-523	0.044	11.860
Tai35a	2422002	0.618	0.579	-2288	0.039	6.311

VII. EXPERIMENTAL RESULTS

Tuned A-TS is compared with a set of the best heuristic methods available for the QAP, such as the genetic hybrid method of Fleurent and Ferland [4] (GH), the reactive tabu search of Battiti and Tecchiolli [2] (RE-TS), the tabu search of Taillard [5] (RO-TS) and a simulated annealing from Connolly [3] (SA). In the comparison, a large subset of well-known problem instances is considered, with sizes between n = 12 and n = 35, contained in the QAPLIB.

The complexity of one iteration, for the compared algorithms, varies: SA has the lower complexity with O(n) per iteration. RO-TS and RE-TS have a complexity of O(n^2) per iteration, GH has a complexity of O(n^3), while A-TS has a complexity of O($n^2/2$) per iteration.

To make fair comparisons between these algorithms, the same computational time was given to each test problem trial, by performing a number of iterations equal to $20nI^{max}$ for A-TS, to $10nI^{max}$ [18] for RE-TS and RO-TS, $125n^2I^{max}$ [18] for SA and $2.5I^{max}$ [18] for GH.

Tests are performed with I^{max}=10 and I^{max}=100. The reason to compare algorithms on short and on long runs is to evaluate their ability in producing relatively good solutions under strong time constraints versus producing very good solutions when more computational resources are available.

Table II compares A-TS with the above-mentioned methods on real life, irregular and structured problems. In particular, the average quality of the solutions produced by these methods is shown, measured in percent above the best solution value known. The RE-TS, S-TS, and RO-TS data contained in table I, II, III, IV and V was gathered from L. M. Gambardella, É. D. Taillard and M. Dorigo [15]. The results of the mentioned authors are averaged over 10 runs, while the results of A-TS are averaged over 100 runs. The experiments evaluate their ability in producing relatively good solutions under strong time constraints.

Table II shows that, methods like RE-TS or SA are not well adapted for irregular problems. Sometimes, they produce solutions over 10% worse than the best solutions known. For problem types *Tai..b*, GH seems to be the best method overall. For problem instances that originate from real life applications (*Kra30a* and *Kra30b*) A-TS performs best. Our approach produces solutions with average deviation smaller than 3% in most of the cases.

TABLE II.
QUALITY OF VARIOUS HEURISTIC METHODS FOR IRREGULAR PROBLEMS AND SHORT RUNS MEASURED IN PERCENT ABOVE THE BEST SOLUTION VALUE KNOWN. BEST RESULTS ARE IN BOLDFACE

Problem name	**Best known value**	**RO-TS**	**RE-TS**	**SA**	**GH**	**A-TS**
Tai20b	122455319	6.700	—	14.392	**0.150**	13.886
Tai25b	344355646	11.486	—	8.831	**0.874**	2.589
Tai30b	637117113	13.284	—	13.515	**0.952**	3.707
Tai35b	283315445	10.165	—	6.935	**1.084**	2.635
Kra30a	88900	2.666	2.155	1.813	1.576	**0.495**
Kra30b	91420	0.478	1.061	1.065	0.451	**0.002**
Chr25a	3796	15.969	16.844	27.139	15.158	**1.452**

Table III provides the same type of comparisons as those of table II, only for regular, unstructured problems. Table III shows that for all of the listed problems, our technique outperforms the other methods. For all of the problem instances the average gap (deviation from the optimal) of the produced solutions is bellow 1%.

TABLE III.
QUALITY OF VARIOUS HEURISTIC METHODS FOR REGULAR PROBLEMS AND SHORT RUNS MEASURED IN PERCENT ABOVE THE BEST SOLUTION VALUE KNOWN. BEST RESULTS ARE IN BOLDFACE.

Problem name	**Best known value**	**RO-TS**	**RE-TS**	**SA**	**GH**	**A-TS**
Nug20	2570	0.101	0.911	0.327	0.047	**0**
Nug30	6124	0.271	0.872	0.500	0.249	**0.020**
Tai20a	703482	0.769	0.705	1.209	0.732	**0.250**
Tai25a	1167256	1.128	0.892	1.766	1.371	**0.814**
Tai30a	1818146	0.871	1.044	1.434	1.160	**0.327**
Tai35a	2422002	1.356	1.192	1.886	1.455	**0.579**

In Table IV and V results obtained with A-TS on longer runs, setting I^{max}=100 are shown.

Comparison for irregular problems and long runs is provided in Table IV. Similarly as in Table II, GH performs best for problem types *Tai..b*. For the other problem instances on long runs our method produces best solutions.

TABLE IV.
QUALITY OF VARIOUS HEURISTIC METHODS FOR IRREGULAR PROBLEMS AND LONG RUNS MEASURED IN PERCENT ABOVE THE BEST SOLUTION VALUE KNOWN. BEST RESULTS ARE IN BOLDFACE

Problem name	Best known value	RO-TS	RE-TS	SA	GH	A-TS
Tai20b	122455319	**0**	—	6.7298	**0**	3.267
Tai25b	344355646	0.0072	—	1.1215	**0**	0.168
Tai30b	637117113	0.0547	—	4.4075	**0.0003**	1.885
Tai35b	283315445	0.1777	—	3.1746	**0.1067**	2.300
Kra30a	88900	0.4702	2.0079	1.4657	0.1338	**0.027**
Kra30b	91420	0.0591	0.7121	0.1947	0.0536	**0**
Chr25a	3796	6.9652	9.8894	12.497	2.6923	**0**

Table V presents the results obtained for long runs on regular problems showing the apparent dominance of A-TS over the other four techniques presented. In half of the cases, our results achieve the exact best solutions in nearly all 100 trials, whereas in the rest, the average gap is below 0.1%.

TABLE V.
QUALITY OF VARIOUS HEURISTIC METHODS FOR REGULAR PROBLEMS AND LONG RUNS MEASURED IN PERCENT ABOVE THE BEST SOLUTION VALUE KNOWN. BEST RESULTS ARE IN BOLDFACE.

Problem name	Best known value	RO-TS	RE-TS	SA	GH	A-TS
Nug20	2570	**0**	0.911	0.070	**0**	**0**
Nug30	6124	0.032	0.872	0.121	0.007	**0**
Tai20a	703482	0.211	0.246	0.716	0.268	**0.003**
Tai25a	1167256	0.510	0.345	1.002	1.189	**0.312**
Tai30a	1818146	0.340	0.286	0.907	0.439	**0.0005**
Tai35a	2422002	0.757	0.355	1.345	0.698	**0.0664**

The above presented results show that for problem types *Tai..b* GH performs best and RO-TS performs slightly worst. On the contrary SA is not well adapted for this kind of problems and RE-TS is the worst method overall. For *Tai..b* problems A-TS shows slightly lower performance than for the other problem types.

Under strong time constraints and for irregular problem instances A-TS and GH seem to be the best methods overall. It is important to mention that, for problem instances that originate from real life applications A-TS produces best quality solutions.

For regular, unstructured problems A-TS outperforms all of the other methods for all of the problem instances considered.

The solutions produced by A-TS for regular, unstructured problems have a slightly better quality than for irregular, structured problems. This can be explained by the fact that relatively good solutions of the unstructured problems are spread in the whole feasible solution set [6]. If all the good solutions are concentrated in a close subset of the feasible solutions, the performance of A-TS slightly decreases. When large number of relatively good solutions are spread all over the solutions space, our method effectively explores the solution space and always finds the optimal solutions. This indicates that A-TS implements good diversification strategy.

Additional comparison of the algorithms, based on the number of iterations needed to achieve the optimal solution, was performed. A series of runs performed with the A-TS were compared with published results of the metaheuristic search techniques RE-TS and RO-TS.

Table VI shows comparisons over some of the problems from the Taillard set, ranging in sizes from 12 to 35. Hundred runs were performed on each problem by A-TS, opposed to 30 runs performed by the authors of the other approaches. The RE-TS and RO-TS data contained in this table was gathered from Battiti and Tecchiolli [2]. The best result in each row is bolded.

TABLE VI.
COMPARISON OF AVERAGE ITERATIONS BEFORE CONVERGENCE TO BEST SOLUTION FOR RE-TS, RO-TS, AND A-TS.

Problem name	Best known value	Average iterations to best solution		
		RE-TS	RO-TS	A-TS
Tai12a	224416	282.3	210.7	**165.7**
Tai15a	388214	**1780.3**	2168.0	2145.5
Tai17a	491812	4133.9	5020.4	**4094.25**
Tai20a	703482	37593.2	34279	**31650.8**
Tai25a	1167256	38989.7	80280.4	**16619.08**
Tai30a	1818146	**68178.2**	146315.7	81038.33
Tai35a	2422002	281334.0	448514.5	**240595.1**

The experimental results presented in Table VI show that our algorithm performs concurrently against other techniques. In nearly all of the cases, A-TS converges in less number of iterations than the other variations of Tabu Search.

VIII. CONCLUSION

This paper describes a novel approach to the Tabu search scheme. We propose a new decision making mechanism using a new evaluation function integrated within the standard TS. The resulting Adaptive Tabu Search (A-TS) augments the exploration and exploitation of the search space, through the incorporation of long-term memory, aspiration criteria and the value of the move in a single evaluation function. Using search history, the adaptive coefficients within the combined evaluation function provide useful feedback to the process. Also, a new approach in coefficients tuning was presented. We propose implementation of a data mining technique for discovering interesting correlations between the coefficients being tuned and the quality of the produced solutions.

Instances of the Quadratic Assignment problem were used for coefficients tuning and quantitative evaluation of the algorithm. The experimental results showed that A-TS performs favorably. In some cases, the optimal result was found in fewer number of iterations than other techniques. For most of the problems, especially for regular problem instances and real life problem instances, A-TS seems to be the best choice.

Although the aim of using a data mining technique was to improve this particular tabu search algorithm, the implications are quite general. The same ideas can easily be adapted and applied to other tabu search algorithms as well. The utilization of an established data mining technique for coefficients tuning significantly improves the hand-tuned algorithm.

Based on the encouraging results, further research of A-TS will be performed. Its implementation to more complex, real life problems, will provide more details of the algorithm quality and advantages.

REFERENCES

[1] R. E. Burkard, S. E. Karisch and F. Rendl, "QAPLIB—A Quadratic Assignment Problem Library," *Journal of Global Optimization*, 10, 391–403, 1997.

[2] R. Battiti, and G. Tecchiolli, "The Reactive Tabu Search," *ORSA Journal on Computing*, 6, 2, 126-140, 1994.

[3] D. T. Connolly, "An Improved Annealing Scheme for the QAP," *Eur. J. Op. Res.*, 46: 93–100, 1990.

[4] C. Fleurent and J. Ferland, "Genetic Hybrids for the Quadratic Assignment Problem," *DIMACS Series in Mathematics and Theoretical Computer Science,* 16: 190– 206, 1994.

[5] E. D. Taillard, "Robust Tabu Search for the Quadratic Assignment Problem," *Parallel Computing*, 17:443-455, 1991.

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

[7] T. C. Koopmans and M. J. Beckmann, "Assignment Problems and the Location of Economics Activities," *Econometrica*, 25: 53–76, 1957.

[8] E. Taillard, "Comparison of Iterative Searches for the Quadratic Assignment Problem," *Location Science*, 3:87-103, 1995.

[9] K. M. Anstreicher, "Recent Advances in the Solution of Quadratic Assignment Problems," *Mathematical Programming*, Series B 97:27-42, 2003.

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

[11] F. Glover, "Tabu Search – Part I," *ORSA Journal on Computing*, 1(3): 109-206, 1989.

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

[13] F. Glover and M. Laguna, "Tabu Search," *Kluwer Academic Publishers*, 1997.

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

[15] T. G. Crainic, M. Gendreau and J. Potvin, "Parallel Tabu Search," 2005, Montreal, Canada.

[16] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," Proc. 20th Int. Conf. *Very Large Data Bases*, 1994.

[17] R. Srikant and R. Agrawal, "Mining Quantitative Association Rules in Large Relational Tables," *SIGMOD* - 1996.

[18] L. M. Gambardella, É. D. Taillard and M. Dorigo, "Ant colonies for the quadratic assignment problem," *IDSIA* - 1997.

[19] G. I. Webb, "Discovering Significant Rules," *KDD*, August 20-23, 2006, Philadelphia, Pennsylvania, USA.