Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 10 (18): 1991-2000, 2010
ISSN 1812-5654

A New Machine Learning based Approach for Tuning Metaheuristics for the Solution of Hard Combinatorial Optimization Problems

M. Zennaki and A. Ech-Cherif
Laboratoire de Modélisation et d'Optimisation des Systèmes Industriels,
Department of Computer Science, Faculty of Science, USTOMB, BP 1505 El-M'naouer-Oran, Algeria

Abstract: This study deals with the problem of tuning metaheuristics for the solution of hard combinatorial optimization problems using machine learning techniques. Decision rules, learned from a corpus of various solutions of randomly generated instances, are repeatedly used to predict solutions quality for a given instance of the combinatorial problem when solved by the metaheuristic. This predicted solution quality is used to fine tune and guide the metaheuristic to more promising search regions during the course of its execution. Results from extensive experimentation on a wide range of hard combinatorial optimization problems ranging from the knapsack problem to the well known Travelling Salesman Problem (TSP) show a noticeable improvement in the objective function value of the solution found by our approach as well as the execution time compared to plain metaheuristics. However, the process of building the corpus and extracting the classification rule is still time consuming but we think it is worth the effort given the fact that this corpus is built only once and also, can provide quick and quality solutions for a stream of instances of the combinatorial problem.

Key words: Heuristics, supervised learning, reactive search, tabu search, support vector machine, multi-class classification

INTRODUCTION

Heuristics remain the only mean to handle large size instances of combinatorial optimization problems and obtain near-optimal solutions in reasonable CPU time. However, the success of such methods depends largely on the correct tuning of some of the metaheuristics parameters such as the prohibition period in tabu search, temperature parameters in simulated annealing etc. The tuning process is usually carried by hand in a trial-and-error procedure guided by some rules of thumb.

Many attempts, based on the reactive search framework, have been made in order to automate the tuning process of some well known metaheuristics for the solution of specific instances of combinatorial optimization problems and some quite encouraging results have been achieved on a wide variety of benchmark problems ranging from max-clique (Battiti and Prostasi, 2001; Battiti and Mascia, 2006), TSP (Battiti and Brunato, 2001, 2005; Breimer *et al*., 2005), linear arrangement problem (Rodriguez-Tello and Hao, 2005), graph matching problem (Sammoud *et al*., 2006), graph similarity (Sorlin and Solnon, 2005) to cooperative vehicles for mars exploration (Williams *et al*., 2001). The reactive search framework is thus defined as metaheuristics in which a feedback on the search is obtained and used to modify some parameters, while the search itself is still in progress. It is worthwhile to emphasize that all the above cited works are based on the resolution of a specific instance of a combinatorial problem, namely the adjustment process of the heuristic parameters is carried out for the instance at hand and this process must therefore be repeated over for different instances of the same problem. Those approaches, which can be considered as online primitive learning techniques, have been focusing on incorporating sophisticated and time consuming adjustment techniques for handling difficult instances which may not be the typical ones encountered in practice.

Recently, some new approaches have been considered for fine tuning metaheuristics parameters which are, based on new machine learning techniques such as neural networks, reinforcement learning, etc. Zhang and Dietterich (1996) use a reinforcement learning method where the value function is approximated with a neural network, for the solution of scheduling problems. (Boyan and Moore, 1997, 2000) developed the stage method is developed which is based on the idea of cpredicting the outcome of a local search on the basis of

Corresponding Author: Laboratoire de Modélisation et d'Optimisation des Systèmes Industriels,
Department of Computer Science, Faculty of Science, USTOMB,
BP 1505 El-M'naouer-Oran, Algeria Tel: 213661206990 Fax: 21341530786

information on its starting point. The prediction is performed using a lazy learning approach. An extension of this approach has been proposed by Moll *et al.* (1999) adding the possibility of refining the information gathered along the search. A related approach is described by Allen and Minton (1996) which aims at predicting the performance of different heuristics on a given instance. The reinforcement learning formalism is adopted also by Miagkikh and Punch (1999) for tackling the Quadratic Assignment Problem. Birkendorf and Simon (1998) describe boost, a method that learns along a search to improve a set of solutions. Moreover, linear regression is used to predict over multiple simulated annealing runs, the long term outcome achieved by starting from a given initial solution (Su *et al.*, 2001).

In a landmark contribution Birattari (2005), gave a formal statement of the tuning problem from a pure machine learning point of view and proposed an algorithm called F-Race for tackling the tuning problem essentially based on an automatic learning procedure for configuring and fine-tuning metaheuristics. A probabilistic measure P_I defined over the space of instances is used to define the expected behavior of a metaheuristic over the class of instances at hand and therefore to select the most appropriate configuration in terms of tuning parameters. This approach is based on a supervised learning of the heuristic parameters over a stream of instances of the problem at hand and the extracted decision rule can be used to effectively tune the heuristic parameters when solving new instances supposed to be drawn from the same probability distribution.

Our approach is similar in spirit to Birattari (2005) in the sense that it aims at solving a stream of instances efficiently rather than focusing on a specific instance, but proceeds differently by learning and predicting solutions quality. This information is used to guide the metaheuristic to more promising regions in order to avoid local optima. To achieve this goal, modern supervised learning techniques are used over a stream of problem instances that should be reasonably considered as representative of the whole class of instances that the algorithm will eventually encounter. The main idea is to build a corpus composed of pairs of examples, made of solutions and problem instance data, along with their quality (ranging from good to bad solutions). The solutions are obtained by solving a set of randomly generated instances using exact or approximate methods. Kernel supervised learning is thus performed on this corpus considered as a training set, to extract the decision rule that can predict the quality of any solution encountered in the course of the metaheuristic execution.

LEARNING FROM DATA

Machine learning is a relatively new promising field and theoretical foundations as well as efficient algorithms have been developed for the last three decades. Particularly, supervised learning (Vapnik, 1998), which plays a major role in our work, can be stated as follows: Given a set of labelled training data or examples Z drawn from an unknown but fixed probability distribution, the task is to construct a decision rule that can predict future examples, called test data, with high probability (Fig. 1).

The training set is $Z = \{(x_1,y_1), (x_2,y_2),\ldots,(x_n,y_n)\}$ where x_i represent the ith example and y_i its label or class. In our case x_i represent the solution concatenated with the instance data and y_i represent the solution quality ranging from good to bad quality.

Using prior assumptions and by induction we can learn an estimated function which can be efficiently used to predict labels of future data (Working or test set). Many approaches have been used to estimate the decision rule such as neural networks, decision trees, linear Fisher discriminant and more recently Kernel Methods and particularly the successful Support Vector Machines (SVM). In the next subsections, a brief introduction to Kernel Methods is given.

Kernel methods: Kernel methods and particularly Support Vector Machines (SVM) (Guyon *et al.*, 1993; Hastie and Tibshirani, 1998; Joachims, 1998), introduced during the last decade in the context of statistical learning (Vapnik, 1979; Vapnik and Tscherwonenkis, 1979; Vapnik and Chervonenkis, 1974), have been successfully used for the solution of a large class of machine learning tasks such as categorization, prediction, novelty detection, ranking and clustering (Boser *et al.*, 1992; Joachims, 1998; Osuna *et al.*, 1997).

In their basic form SVMs are used in two-class supervised learning, where labels of examples (quality of solutions in our case) are known to take only two values. Given a sample of n training examples $Z=\{(x_1,y_1),\ldots,(x_n,y_n)\}$, $x_i \in IR^p$, $y_i \in \{-1,+1\}$, Linear SVM finds a decision rule in the form of a hyperplane which

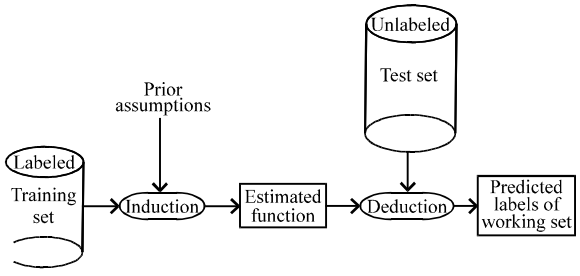

Fig. 1: Supervised learning

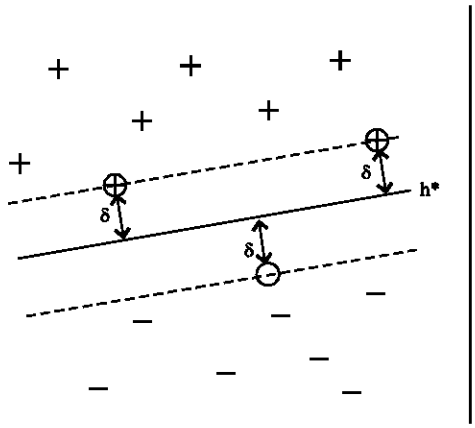

Fig. 2: Optimal plane h* maximizes margin δ

maximizes the Euclidian distance to the closest training examples, this distance is called the marginδ, as shown in Fig. 2 (Joachims, 1998).

The SVM, in their general form, extend an optimal linear decision rule or hypothesis, in terms of an upper bound on the expected risk that can be interpreted as the geometrical margin, to non linear ones by making use of kernels k (.,.). Kernels represent dissimilarity measures of pairs of objects in the training set Z. In standard SVM formulations, the optimal hypothesis sought is of the form Eq. 1.

$$f(x) = \sum c_i k(x, x_i) \tag{1}$$

where c_i are the components of the unique solution of a linearly constrained quadratic programming problem, whose size is equal to the number of training patterns. The solution vector obtained is generally sparse and the non zero c_i's are called Support Vectors (SV's). Clearly, the number of SV's determines the query time which is the time it takes to predict novel observations and subsequently, is critical for some real time applications such as speech recognition tasks.

It is worth noting that in contrast to connectionist methods such as neural networks, the examples need not have a Euclidean or fixed-length representation when used in kernel methods. The training process is implicitly performed in a Reproducing Kernel Hilbert Space (RKHS) in which k(x;y) is the inner product of the images of two example x, y. Moreover, the optimal hypothesis can be expressed in terms of the kernel that can be defined for non Euclidean data such biological sequences, speech utterances etc. Popular positive kernels include (2, 3, 4, 5):
Linear

$$k(x_i, x_j) = x_i^T x_j \tag{2}$$

Polynomial

$$k(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0 \tag{3}$$

Gaussian

$$k(x_i, x_j) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right), \gamma > 0 \tag{4}$$

Laplacian

$$k(x_i, x_j) = \exp\left(-\gamma \sum_k \left| x_{ik} - x_{jk} \right|\right), \gamma > 0 \tag{5}$$

The kernel function represents the similarity of examples into that space. This notion confers to SVMs an important flexibility, a reasonable CPU time and an ability to generalize even in the presence of an important number of features. Figure 3 shows how a kernel function can project the initial set of examples (Fig. 3a) into a large dimension space (Fig. 3b) in order to allow linear separation.

SVM formulation and training: Given training vectors $x_i = \epsilon \Re^n$ i = 1,....., m, in two classes and a vector $y \epsilon \Re^m$ such that, $y_i \in \{1,-1\}$, Support Vector Classifiers (Guyon *el al.*, 1993; Joachims, 1998; Vapnik, 1998; Vapnik and Chervonenkis, 1974) solve the following linearly constrained convex quadratic programming problem (6):

$$\begin{aligned} &\text{maximize} \quad W(a) = \frac{1}{2}\sum_{i,j} a_i a_j y_i y_j k(x_i, x_j) - \sum_{i=1}^{m} a_i \\ &\text{subject to the constraints:} \quad \forall i,\ 0 \le \alpha_i \le C \\ &\qquad\qquad \sum_{i=1}^{m} a_i y_i = 0 \end{aligned} \tag{6}$$

The optimal hypothesis is (7):

$$f(x) = \sum_{i=1}^{m} \alpha_i y_i k(x, x_i) + b \tag{7}$$

where, the bias term b can be computed separately (Osuna *et al.*, 1997). Clearly, the hypothesis f depends only on the non null coefficients α_i whose corresponding patterns are called Support Vectors (SV).

The QP objective function involves the problem Gram matrix K whose entries are the similarities $k(x_i, x_j)$between the patterns x_i and x_j. It is important to note, on one hand, that the pattern input dimentioned, in the above formulation, is implicit and does not affect to some extent the complexity of training, provided that the Gram matrix K can be efficiently computed for the learning task at hand. On the other hand, the patterns representation is

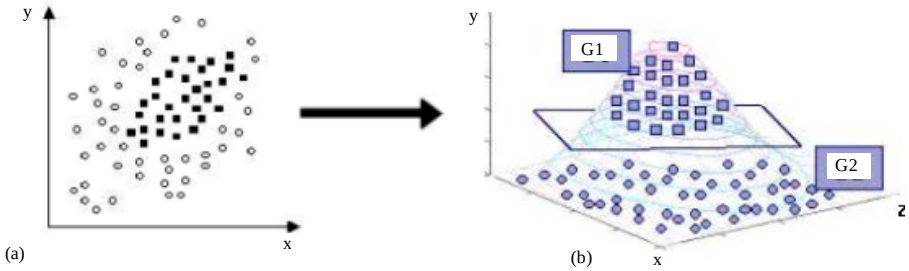

Fig. 3: Kernel functions allow linear separation of two classes (a) G1 and (b) G2

not needed and only pair wise similarities between objects must be specified. This feature makes SVM very attractive for high input dimensional recognition problems and for the ones where patterns can not be represented as fixed dimensional real vectors such as text, strings, DNA etc. For large scale corpora however, the quadratic programming problem becomes quickly computationally expensive, in terms of storage and CPU time. It is well known that general-purpose QP solvers scale with the cube of the problem dimension which is, in our case, the No. of training examples m. Specialized algorithms, typically based on gradient descent methods, achieve impressive gains in efficiency, but still become impractically slow for problems whose size exceeds 100,000 examples. Several attempts have been made to overcome this shortcoming by using heuristically based decomposition techniques such as Sequential Minimal Optimization SMO (Osuna *et al*., 1997) implemented in LibSVM package (Chang and Lin, 2001).

Multiclass extensions: In our case, we have many classes, each class represent a solution quality. However Support Vector Machines are inherently binary classifiers and its efficient extension to multiclass problems is still an ongoing research issue (Dietterich and Bakiri, 1995; Duan and Keerthi, 2003; Guyon *et al*., 1993; Hastie and Tibshirani, 1998; Hsu and Lin, 2002; Platt *et al*., 1999; Rifkin and Klautau, 2004). Several frameworks have been introduced to extend SVM to multiclass contexts and a detailed account of the literature is out of the scope of this study. Typically multiclass classifiers are built by combining several binary classifiers as depicted in Fig. 4 (Hsu and Lin, 2002). The earliest such method is the one-against-all (OVASVM) (Rifkin and Klautau, 2004) which constructs K classifiers, where K is the number of classes. The k^{th} classifier is trained by labeling all the examples in the k^{th} class as positive and the remainder as negative. The final hypothesis is given by the formula (8):

$$f_{ova}(x) = \arg\max_{i=1,\ldots,k}(f_i(x)) \quad (8)$$

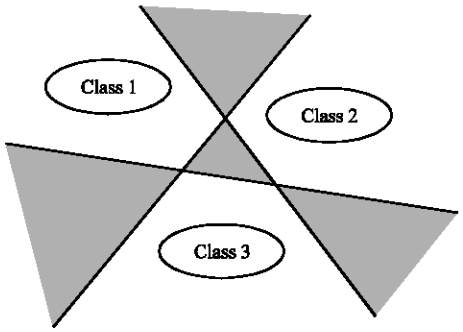

Fig. 4: Multi-class SVM

Another popular paradigm, called one-against-one, proceeds by training

$$\frac{k(k-1)}{2}$$

binary classifiers corresponding to all the pairs of classes. The hypothesis consists of choosing either the class with most votes (voting) or traversing a directed acyclic graph where each node represents a binary classifier (DAGSVM) (Platt *et al*., 1999). There was debate on the efficiency of multiclass methods from statistical point of view. Clearly, voting and DAGSVM are cheaper to train in terms of memory and computer speed than OVASVM. Hsu and Lin (2002) investigated the performance of several SVM multi-class paradigms and found that the one-against-one achieved slightly better results on some small to medium size benchmark data sets.

GENERAL METHODOLOGY FOR INCORPORATING ACQUIRED KNOWLEDGE INTO METAHEURISTICS

Given a metaheuristic for solving a hard combinatorial problem, our approach consists in predicting solution quality using decision rules learned beforehand from a constructed corpus of examples. It proceeds by first

building a large corpus of examples, consisting of pairs of solutions and their quality and then learns a decision rule which permits to predict with a high accuracy the quality of solutions. Under reasonable assumptions relating to the learnability of such decision rules and that the constructed corpus is representative of all the instances to be solved, this acquired external knowledge will allow the metaheuristic to efficiently choose the most promising neighbourhood to explore next and therefore avoid to be trapped in local optima.

General assumptions: For our approach to be valid for efficiently tuning a metaheuristic, designed to solve a stream of instances which are assumed to be drawn iid from a distribution of a hard combinatorial problem instance data, we make use of the following assumptions:

Instance sampling: We assume that the instance data we use in building the corpus are drawn identically and independently from a fixed but unknown distribution. To ensure this, we sample the coefficients of the combinatorial problem randomly using a uniform random number generator. For each generated instance, the metaheuristic is run and all the solutions visited by the metaheuristic are concatenated with the instance data to form a set of examples whose labels are categorized into classes according to their closeness to the final optimal or near optimal solution found by the metaheuristic.

Optimal solution quality: We assume that the sampled instances can be solved exactly to optimality or near optimality using exact or eventually approximate methods. It is worthwhile to mention here that only an exact or good estimate of the optimal objective function value is needed rather than the optimal solution itself. This will ensure that the label associated with the example is accurate. It is important to mention here also that on one hand, learning with approximate labels has been studied in the literature (Tao *et al*., 2004) and on the other hand, that approximate polynomial schemes have been developed for solving large instances of hard combinatorial problems which ensure a high percentage of the optimal solution (Baker, 1994; Shmoys *et al*., 1997).

Class representativity of the corpus: We assume that the set of examples in the training corpus is fairly and equally distributed among all the classes considered ranging from good to bad quality.

Nature of the problems to be solved: Finally, we are assuming that we are dealing with a stream of instances whose data follow a fixed but unknown distribution rather than solving a specific isolated instance. This assumption can be argued to be a reasonable one in many practical contexts. For instance, in Vehicle Routing Problems (VRP), which have to be solved in e-commerce, one always assume that the demands are random and thus most of the instances can be reasonably considered being generated by some probability distribution which reflects the pattern of demands for a given region.

Under the above assumptions, general conditions on the generalization ability of the constructed learned rule have been established (Poggio *et al*., 2004). Those conditions have been verified to a large extent in our case. We discuss this issue in the experimental part of the study.

Corpus building: In order to extract the decision rule for predicting the quality of unseen or novel encountered solutions in the course of executing a given metaheuristic, our approach consists in building a training corpus composed of pairs of examples, made of solutions and problem instance data, along with their quality from randomly generated instances of the combinatorial problem at hand. In the process of constructing such corpora, each instance is solved to optimality or near optimality and the quality of solutions is evaluated according to their closeness to the optimal solution found. When solving problem instances, we consider optimal solutions and also others ones. This set of solutions is classified into categories ranging from optimal or good solutions to bad solutions. In the case of large instances intractable by exact methods, we use an approximate resolution. We called the training set thus obtained an approximate corpus rather than exact one (obtained when using exact methods).

Learning the decision rule: In order to construct the decision rule, LIBSVM package (Chang and Lin, 2001), considered as the state of the art SVM implementation package, has been used in our implementation. The training corpus built is structured in a manner that the example label corresponds to the solution quality and the example detail to the instance description concatenated with the solution vector.

An example of TSP instances: The structure of constructed training corpus allows direct use of multi-class Support Vector machines.

This example (Fig. 5) describes first the solution quality ranging from 1 to 10; 1 means optimal or good solution, 10 a bad solution. Then the instance detail is given. In the case of TSP, the problem is represented by the incidence matrix of the complete graph, in which the

Solution Class.....	Instance description.....	Solution detail
1	1:45 2:76 3:86	51:1 52:0 53:1 .....
2	1:45 2:76 3:86	51:1 52:1 53:0 .....
2	1:45 2:76 3:86	51:1 52:0 53:0 .....
.....		
.....		
10	1:45 2:76 3:86	51:1 52:0 53:0 .....
1	1:40 2:90 3:74	100:1 101:1.....
.....		

Fig. 5: Training dataset structure

Hamiltonian path will be found. Depending on the size of the instance, the solution detail corresponding to a binary vector is given after; in the first instance from dimension 51 and for the second instance from 100.

Multi-class supervised learning: The training set is built as LIBSVM benchmarks structure (Chang and Lin, 2001) used in machine learning, like described before, to allow direct use of Multi-class SVM. A supervised learning is thus performed on the build corpus considered as a training data set. We use for the Multi-class classification, the publicly available program UniverSVM (Sinz, 2006) with polynomial kernel which gives us the best results of prediction.

Using predicted solution quality to guide the metaheuristic: Finally we use the model or the discriminant function obtained by Multi-class SVM for guiding metaheuristics which will be informed of the quality of the solution at hand while solving new instances intractable by exact method or in the case where not enough CPU time is available for an exact resolution of the instance.

ILLUSTRATING EXAMPLE

From a theoretical point of view, the proposed approach seems to be limited because doing an automatic learning will be efficient only if we can achieve an exact resolution on an important account of instances (even if approximate training dataset gave us acceptable results). Building such corpora seems consuming time as building human genome!

On the contrary, from a practical point of view, the instances have generally common characteristics. In this case, machine learning can be very efficient by guiding the resolution process of new instances. We can consider as concrete example inspired from Birattari (2005) an important distribution company of goods over a city to its many customers. Every day, the company must answer to customer commands which are collected in the precedent day. The important number of commands yields to a difficult scheduling of the distribution process. Every day, the company is thus confronted to solving a TSP instance (known as one of the hardest NP-complete problem). Furthermore, this example is characterized by a stream of instances observed by the company.

It should be noticed that among all possible TSP instances, some of them will be more likely to occur than others. For example, the morphological features of the city itself rules out a large share of possible instances: An instance presenting nodes that fall outside the urban area will never occur. The same applies if a node of the instance is placed in a non residential or industrial area or generally in some special regions where no customer will ever request a delivery. On the contrary, an instance is more likely to occur in residential and industrial areas. Similarly, if the company has advertised in some neighbourhoods, it is to be expected that instances that present nodes in these neighbourhoods will occur more likely. From this we can see that in practice, the instances share a common morphology and a supervised learning over a stream of such instances can be very profitable to the resolution of future instances. The link between the different phases of our approach is therefore ensured by the hypothesis that the future will be in some sense similar to the past. Furthermore the preliminaries results we obtained from randomly generated instances encourage us to continue in this sense included those results obtained with training sets generated by an approximate resolution (approximate corpus).

Table 1: Results using exact training dataset (by prediction rate)

Class	1	2	3	4	5	6	7	8	9	10
1	85.14	10.45	4.41	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	14.31	82.99	2.70	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	7.32	91.44	1.24	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	98.12	1.88	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	5.44	91.11	3.45	0.00	0.00	0.00	0.00
6	0.00	0.00	4.43	6.34	10.34	75.71	3.18	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00	13.44	20.44	65.45	0.67	0.00	0.00
8	0.00	0.00	0.00	0.00	1.44	13.23	25.45	56.23	3.65	0.00
9	0.00	0.00	0.00	0.00	1.23	9.23	11.24	25.45	50.45	2.40
10	0.00	0.00	0.00	0.00	0.00	4.32	14.54	35.45	40.45	5.24

Table 2: Results using approximate training dataset (by prediction rate)

Class	1	2	3	4	5	6	7	8	9	10
1	8.26	11.93	19.27	48.62	10.09	1.83	0.00	0.00	0.00	0.00
2	2.78	10.83	25.56	31.94	26.11	1.39	0.56	0.00	0.56	0.28
3	3.97	6.61	20.37	27.25	27.78	8.73	2.91	1.59	0.53	0.26
4	3.32	5.05	15.96	26.33	26.06	15.82	4.79	1.20	0.80	0.66
5	1.32	2.03	9.14	22.13	29.44	20.71	10.36	2.34	1.42	1.12
6	1.08	1.22	5.28	16.53	24.80	28.86	13.14	5.01	1.63	2.44
7	0.00	0.55	2.49	9.42	22.44	34.63	21.88	5.26	1.39	1.94
8	0.00	0.00	2.22	3.70	14.81	36.30	29.63	10.37	0.74	2.22
9	0.00	0.00	0.00	5.26	15.79	15.79	42.11	15.79	5.26	0.00
10	0.00	0.00	0.00	1.96	13.73	37.25	33.33	13.73	0.00	0.00

EXPERIMENTS

All the algorithms developed in this study have been implemented in C++ Language using a workstation Hewlett-Packard bi-processor XW-6000, 1.40 MHZ per processor and 2Go RAM in the Laboratoire de Modélisation et d'Optimisation des Systèmes Industriels (LAMOSI)-USTO Oran.

For the supervised learning, we generate randomly about 125 instances of TSP composed of 20 nodes. The distance between nodes is generated randomly between 1 and 30 km. for each instance we keep about 150-200 solutions in the corpus which represent the training set. The solutions are classified from 1 to 10, namely from optimal solutions to bad ones. We use for exact resolution a simple branch and bound procedure.

For the Multi-class classification, we use the publicly available program UniverSVM (Sinz, 2006) with polynomial kernel which gave us the best results of prediction. Finally, a test dataset is generated like train dataset and used to compare the solutions quality with predictions obtained by the decision rule obtained in the learning phase.

All the obtained results are summarized in Table 1 called quality matrix which indicates the prediction rate of each class. For example the prediction rate of the class 1 of optimal and good solutions is 85.17, namely a given solution is recognized to be optimal in 85.17%. Even if the instances are generated randomly, the results are very interesting. In fact, a good solution is considered as a good solution in about 90% of all the cases and a bad solution is considered as a bad one in about 80% of all the cases. We are convinced that if the instances come from a real case, the results will be better. We can see that a perfect quality matrix is diagonal.

In the other hand and in order to deal with large size problems, we perform an approximate resolution of instances for generating an approximate training dataset. For this, we also generate randomly about 125 instances of TSP composed of 50 nodes. The results of prediction using an approximate training dataset are summarized in Table 2 and shows a logically worst results compared to those obtained using an exact training dataset. In fact, a good solution is considered as a good solution in about 78% of all the cases and a bad solution is considered as a bad one in about 69% of all the cases. However, in this case also, if the instances come from a real case, the results will be better.

Similar results are obtained with other combinatorial problems such as Knapsack problem, set covering problem and vehicle routing problem. Once such corpora built (one for each combinatorial problem), they can be used after a supervised learning, for fine tune metaheuristics and guide the search process to more promising regions. We propose in the next section an integration of this acquired knowledge to the tabu search metaheuristic.

APPLICATION TO TABU SEARCH METAHEURISTIC

we show how we can integrate the prediction of solutions quality to metaheuristics by guiding the search process to more promising regions and/or, to adjust heuristic parameters while solving other instances.

We propose a modified general scheme of the tabu search metaheuristic (Glover 1989, 1990) by incorporating the predicted solution quality while solving new instances:

Initialization:
Given an initial solution $x_0 \in X$
$\hat{Z} = Z(x_0)$
k = prohibition period representing the tabu list size
At Step n:
Given $x_n \in X$, the current solution;
Find in a sub-neighborhood N* of $N(x_n)$ (called candidate list) the best solution x* which is:
- non tabu or
- tabu, but satisfying the aspiration criterion
Set $x_{n+1} = x^*$
If $Z(x^*) < \hat{Z}$ then $\hat{Z} = Z(x^*)$
Update the tabu list
Switchdepending on the predicted quality of x_{n+1}:
Case quality of $x_{n+1} = 1$: Stop the algorithm
Case quality of x_{n+1} between 2 and 4 : Intensification of the search
Case quality of $x_{n+1} > 4$: Perform a descent procedure followed by a diversification of the search in the case where the solution quality remain > 4.

The simple idea we propose in this modified algorithm is to use the predicted current solution quality as a stop criterion in the case where the solution is estimated to be of high quality (Class 1). If we think that the current solution is good (predicted quality between 2 and 4), an intensification procedure is performed in order to explore deeply the current region. On the contrary, if the solution quality is estimated mediocre or bad, we perform a descent in order to reach eventually a local optimum. If the quality of this local optimum is bad, a diversification procedure is performed in order to guide the search to more promising regions.

To examine the impact of incorporating the predicted quality solutions we compare the above algorithm to plain tabu search. The combinatorial problem we consider in our experimentation is the well known Vehicle Routing Problem VRP (Dantzig and Ramser, 1959). Among 56 problems of the OR-Library (Beasley, 1996) we consider 10 problems. This library has been used in many experimental studies (Liu and Shen, 1999; Sadouni, 2006).

Each problem instance consists of 100 customers distributed in a square [0,100]. The depot is located at the center of the square. It worth noting that there is three classes of problems in the library depending on the distribution of customers:

- The R Class corresponds to a uniform distribution of customers in the square
- The C Class corresponds to a distribution of customers into regions (clusters) two by two
- The RC Class is a combination of the two precedent classes. A part of customers is uniformly distributed into the square while the other part is distributed into regions

Table 3: Experimentation results

Problem	Tabu search	Tabu search with prediction
1	5061	5061
2	5013	4998
3	4772	4774
4	4455	4358
5	9272	9261
6	8433	8450
7	8033	8033
8	7384	7380
9	5687	5685
10	5649	5649

The 10 problems are selected among the three classes.

The results of our experimentation compared to the classical tabu search metaheuristic are summarized in Table 3 (Sadouni, 2006) and show a noticeable improvement of the objective function value.

CONCLUSIONS

Although, the centrality of tuning is acknowledged in the literature, the problem of tuning metaheuristics has received little attention of researchers which try for the most of them to adjust automatically metaheuristic parameters while solving a specific instance. We propose in this paper a new machine learning based approach for tuning metaheuristics for the solution of combinatorial optimization problems which consist to inform the metaheuristic of the quality of the solution at hand. To achieve this goal, modern supervised learning techniques are used over a stream of problem instances that should be reasonably considered as representative of the whole class of instances that the algorithm will eventually encounter. Decision rules are learned from training corpora allowing the prediction of solution quality. We propose also a modified general scheme of tabu search metaheuristic by incorporating the predicted solution quality allowing automatic oscillation between intensification and diversification search in order to avoid to be trapped in local optima. The encouraging results we obtain confirm our conviction that modern machine learning techniques can be profitable for the combinatorial optimization. However, the process of building the corpus and extracting the classification rule is still time consuming but we think it is worth the effort given the fact that this corpus is built only once for all instances.

As a perspective, we are thinking about using semi-supervised learning which can reduce the CPU time amount of building training corpora. This kind of learning can extract decision rules using both labelled and unlabeled data and has been used successfully in many practical domains where unlabeled data are abundant but labelled data are expensive to generate. This idea will be

beneficial in our case, since training corpora will be build from both labelled or evaluated solutions (requiring exact or approximate resolution) and unlabelled solutions easy to generate.

REFERENCES

Allen, J.A. and S. Minton, 1996. Selecting the Right Heuristic Algorithm: Runtime Performance Predictors. In: Advances in Artificial Intelligence: The Eleventh Biennial Conference of the Canadian Society for Computational Studies of Intelligence, McCalla, G. (Ed.). Springer-Verlag, Germany, Berlin, pp: 41-53.

Baker, B.S., 1994. Approximation algorithms for NP-complete problems on planar graphs. J. Assoc. Comput. Machinery, 41: 153-180.

Battiti, R. and M. Brunato, 2001. Reactive Search for Traffic Grooming in WDM Networks. In: Evolutionary Trends of the Internet, Palazzo, S. (Ed.). Springer, Mexico.

Battiti, R. and M. Prostasi, 2001. Reactive local search for the maximum clique problem. Algorithmica, 29: 610-637.

Battiti, R. and M. Brunato, 2005. Reactive Search: Machine Learning for Memory-Based Heuristics. Trento University, Italy.

Battiti, R. and F. Mascia, 2006. Reactive and Dynamic Local Search for Max-Clique: Does the Complexity Pay Off?. Trento University, Italy.

Beasley, J.E., 1996. Obtaining test problems via internet. J. Global Optimization, 8: 429-433.

Birattari, M., 2005. The problem of tuning metaheuristics as seen from a machine learning perspective. Ph.D. Thesis, Universite Libre de Bruxelles.

Birkendorf, A. and H.U. Simon, 1998. Using computational learning strategies as a tool for combinatorial optimization. Ann. Math. Artifi. Intell., 22: 237-257.

Boser, B.E., I.M. Guyon and V.N. Vapnik, 1992. A training algorithm for optimal margin classifiers. Proceedings of the 5th Annual Workshop on Computational Learning Theory, Pittsburgh, Pennsylvania, United States, July 27-29, ACM, New York, USA., pp: 144-152.

Boyan, J. and A. Moore, 1997. Using prediction to improve combinatorial optimization search. Proceedings of the 6th International Workshop on Artificial Intelligence and Statistics, (AIS'97), Fort Lauderdale, Florida, pp: 311-318.

Boyan, J. and A. Moore, 2000. Learning evaluation functions to improve optimization by local search. J. Machine Learning Res., 1: 77-112.

Breimer, E., M. Goldberg, D. Hollinger and D. Lim, 2005. Discovering optimization algorithms through automated learning. Graphs Discovery, 69: 7-27.

Chang, C.C. and C.J. Lin, 2001. LIBSVM: A library for support vector machines. Software. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Dantzig, G.B. and J.H. Ramser, 1959. The truck dispatching problem. Manage. Sci., 6: 80-91.

Dietterich, T. and G. Bakiri, 1995. Solving multiclass problem via error-correcting output code. J. Artifi. Intell. Res., 2: 263-286.

Duan, K.B. and S.S. Keerthi, 2003. Which is the Best Multiclass SVM Method? An Empirical Study. National University of Singapore, Singapore.

Glover, F., 1989. Tabu search-part I. ORSA J. Comput., 1: 190-209.

Glover, F., 1990. Tabu search-Part II. ORSA J. Comput., 2: 4-32.

Guyon, B., I. Boser and V. Vapnik, 1993. Automatic capacity tuning of very large VC-dimension classifiers. Adv. Neural Inform. Process. Syst., 5: 147-155.

Hastie, T. and R. Tibshirani, 1998. Classification by pairwise coupling. Ann. Statistics, 26: 451-471.

Hsu, C.W. and C.J. Lin, 2002. A comparison of methods for multi-class support vector machines. IEEE Trans. Neural Networks, 13: 415-425.

Joachims, T., 1998. Making Large-Scale Support Vector Machine Learning Practical. In: Advances in Kernel Methods: Support Vector Machines, Scholkopf, B., C. Burges and A. Smola (Eds.). MIT Press, Cambridge, MA., pp: 386.

Liu, F.H. and S.Y. Shen, 1999. A method for vehicle routing problem with multiple vehicle types and time windows. Proc. Natl. Sci. Counc. ROC (A), 23: 526-536.

Miagkikh, V.V. and W.F. Punch, 1999. Global search in combinatorial optimization using reinforcement learning algorithms. Proceedings of the Congress in Evolutionary Computation, (CEC'99), IEEE Publications, Piscataway, NJ, USA., pp: 189-196.

Moll, R., A.G. Barto, T.J. Perkins and R.S. Sutton, 1999. Learning instance-independent value functions to enhance local search. Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II, (CANIPS'99), MIT Press, Cambridge, MA, USA., pp: 1017-1023.

Osuna, E., R. Freund and F. Girosit, 1997. Training support vector machines: An application to face detection. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 17-19, IEEE Computer Society, pp: 130-136.

Platt, J., N. Cristianini and J. Shawe-Taylor, 1999. Large margin DAGs for multiclass classification. Proc. Neural Inform. Process. Syst., 12: 547-553.

Poggio, T., R. Rifkin and S. Mukherjee, 2004. General conditions for predictivity in learning theory. Nature, 428: 419-422.

Rifkin, R. and A. Klautau, 2004. In defense of one-vs-all classification. J. Machine Learning Res., 5: 101-141.

Rodriguez-Tello, E. and J.K. Hao, 2005. Recherche Tabou Reactive Pour le Probleme de L'arrangement Lineaire Minimum. LERIA, Universite d'Angers, France, Tours, pp: 314-315.

Sadouni, K., 2006. Heterogeneous fleet vehicle routing problem with time windows and nonlinearly penalized delays. J. Applied Sci., 6: 1969-1973.

Sammoud, O., S. Sorlin, C. Solnon and K. Ghredira, 2006. A comparative study of ant colony optimization and reactive search for graph matching problems. Proceedings of 6th European Conference on Evolutionary Computation in Combinatorial Optimization, Feb. 28, Springer, Berlin, Heidelberg, pp: 234-246.

Shmoys, D.B., E. Tardos and K. Aardal, 1997. Approximation algorithms for facility location problems. Proceedings of the 29th Annual ACM Symposium on Theory of Computing, May 1997, ACM Press, pp: 265-274.

Sinz, F., 2006. Support vector machine with large scale CCCP functionality. UniverSVM, Version, pp: 1.

Sorlin, S. and C. Solnon, 2005. Reactive tabu search for measuring graph similarity. Proceedings of 5th IAPR Workshop on Graph based Representations in Pattern Recognition, March 10, Springer, Berlin, Heidelberg, pp: 172-182.

Su, L., W.L. Buntine, R. Newton and B.S. Peters, 2001. Learning as applied to stochastic optimization for standard-cell placement. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp: 20.

Tao, Q., S. Scott, N.V. Vinodchandran and T.T. Osugi, 2004. SVM-based generalized multiple-instance learning via approximate box counting. Proceedings of the 21st International Conference on Machine Learning, July 4-8, Banff, Alberta, Canada, pp: 101-101.

Vapnik, V. and A. Chervonenkis, 1974. Theory of Pattern Recognition. Nauka, Moscow, (In Russian).

Vapnik, V., 1979. Estimation of Dependences Based on Empirical Data. Springer-Verlag, Moscow.

Vapnik, W. and A. Tscherwonenkis, 1979. Theorie der Zeichenerkennung. Akademie-Verlag, Berlin.

Vapnik, V.N., 1998. Statistical Learning Theory. Wiley, New York.

Williams, B.C., P. Kim, M. Hofbaur, J. How and J. Kennell *et al*., 2001. Model-based reactive programming of cooperative vehicles for mars exploitation. Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space. http://en.scientificcommons.org/42216072.

Zhang, W. and T.G. Dietterich, 1996. High-Performance Job Shop Scheduling with a Time-Delay TD(ë) Network. In: Advances in Neural Information Processing Systems 8, Touretzky, D.S., M.C. Mozer and M.E. Hasselmo (Eds.). MIT Press, Cambridge, MA, USA., pp: 1024-1030.