Chapter X

TUNING TABU SEARCH STRATEGIES VIA VISUAL DIAGNOSIS

Steven HALIM[1] and Hoong Chuin LAU[2]
[1]*stevenha@comp.nus.edu.sg, School of Computing, National University of Singapore;*
[2]*hclau@smu.edu.sg, School of Information Systems, Singapore Management University*

Abstract: While designing working metaheuristics can be straightforward, tuning them to solve the underlying combinatorial optimization problem well can be tricky. Several tuning methods have been proposed but they do not address the new aspect of our proposed classification of the metaheuristic tuning problem: tuning search strategies. We propose a tuning methodology based on *Visual Diagnosis* and a generic tool called *Visualizer for Metaheuristics Development Framework* (V-MDF) to address specifically the problem of tuning search (particularly Tabu Search) strategies. Under V-MDF, we propose the use of a Distance Radar visualizer where the human and computer can collaborate to diagnose the occurrence of negative incidents along the search trajectory on a set of training instances, and to perform remedial actions on the fly. Through capturing and observing the outcomes of actions in a Rule-Base, the user can then decide how to tune the search strategy effectively for subsequent use.

Key words: Metaheuristics, Software Framework, Tuning Problem, Visualization

1. INTRODUCTION

Metaheuristics have been used extensively to solve hard combinatorial optimization problems, often with significant success. Given that metaheuristics do not guarantee optimality in general, the challenge is not so much to design a *working* algorithm but to tune it so as to obtain the *best* possible result. One way to measure the goodness of a metaheuristic algorithm is by checking its result against a set of *benchmark* problem instances.

Since different problems or even instances of the same problem may require the metaheuristic algorithm to be configured with different search parameters, components and/or strategies in order to work optimally, some resort to trial-and-error tuning through extensive experiments. Others use their past knowledge or experiences to tune the algorithm. From the industry standpoint, this process is unproductive especially against a backdrop of tight development schedules.

Alternatively, human[1] intelligence and machines can collaborate to shorten development time through the use of a well-designed visualization and interaction tool. The human-plus-computer collaboration has obtained considerable success in solving complex tasks, e.g. CAD/CAM. With the help of a well-designed visual diagnostic tool, an algorithm designer is able to examine search trajectories more systematically, steer the search, and readily see the impact of his action. We argue that this significantly reduces the time to design good search strategies which in turn speed up the overall development time.

Using visualization to assist optimization has been proposed in the seminal work of (Jones, 1996). In this paper, we propose a visualization scheme that determines quickly a set of rules that are helpful to the underlying metaheuristic algorithm. Unlike works due to (Klau *et al.*, 2002) and others which focused on *problem-specific* visualization, we emphasize the design of a *generic* problem-independent tool called *Visualizer for Metaheuristics Development Framework* (*V-MDF*). This work is an extension of *MDF* proposed in (Lau *et al.*, 2004b, 2006).

Instead of relying on specific problem domain information, V-MDF seeks to capture a pictorial view of the search trajectories and reports any anomalies to the human user. By visual inspection of these anomalies, the user can determine with higher accuracy the problems encountered during search, and apply remedial actions (such as tuning the parameters, adjusting the components of metaheuristics, or deriving better adaptive search strategies). With V-MDF, the algorithm designer begins with a metaheuristic on some defined search strategies, observes the search run-time dynamics, and dynamically improves the search strategies.

V-MDF differs from existing approaches for tuning metaheuristic which focused on the design of an efficient method for automatically choosing the best parameter values and/or metaheuristic components in black-box fashion (Adenso-Diaz and Laguna, 2006; Birattari, 2004). Instead, we extend the idea of visualizing the search process by (Kadluczka *et al.*, 2004) and that of analyzing the search landscape by (Fonlupt *et al.*, 1999; Merz, 2000; Hoos and Stuetzle, 2005), to help the users in designing better metaheuristics. This feature makes V-MDF especially useful for designing metaheuristics for new combinatorial optimization problems where search strategies have not been well-defined.

[1] The terms *human, user, or algorithm designer* are used interchangeably to refer to those who specialize in the development of metaheuristic algorithms.

This paper proceeds as follows: In Section 2, we discuss metaheuristic tuning problem in a broader sense. In Section 3, we review several tuning methods in the literature and classify them appropriately. In Section 4, a Visual Diagnosis Tuning methodology is proposed, followed by a discussion of V-MDF, the tool to support this methodology, in Section 5. A case study of the usage of V-MDF is given in Section 6. Section 7 gives the conclusions and future directions.

2. THE METAHEURISTICS TUNING PROBLEM

Recently, there is a growing interest in addressing the metaheuristics tuning problem. There are on-going discussions in the literature about the proper definition and scope of the tuning problem, e.g. (Birattari, 2004), as well as various proposals of tuning methods. However, several aspects are often overlooked. In this section, we propose a new classification to put into perspective our broader view of the metaheuristic tuning problem, especially in tuning search strategies.

2.1 Different Types of Tuning Problem

The term ‘tuning’ is often too broad. In the context of metaheuristics, we classify tuning problem into three types.

2.1.1 Type-1: Calibrating Parameter Values

In this ‘easiest’ type of tuning problem, the metaheuristic algorithm has been completely defined; and all the designer needs to do is to set the appropriate parameter values, e.g. setting the tabu tenure, setting the size of candidate list, etc. Different parameter values may influence the overall metaheuristic performance. Seemingly easy as it sounds, the challenge is that varying the value of one parameter may affect the optimal setting of the other parameter values, since the parameters are often correlated. Furthermore, in many practical situations, the range of parameter values is too large for the algorithm designer to determine their values through trial-and-error.

2.1.2 Type-2: Choosing Best Components

In this type of tuning problem, the algorithm designer needs to choose several components that will be used in a particular metaheuristic algorithm, e.g. choosing neighborhood (2/3/k-Opt), tabu list (tabu move/attribute), etc. Typically, each choice of metaheuristic component has its own strengths and weaknesses. (Charon and Hudry, 1995) show that different components have different effects to the performance of metaheuristics. Finding the optimal mix of components of the metaheuristic is often a challenging task as one needs to try a large number of combinations. This type of tuning problem is considered to be more complex than type-1, because once a good configuration is found, one may still need to properly set the parameter values of the components in the chosen configuration.

2.1.3 Type-3: Tuning Search Strategies

In this type of tuning problem, the algorithm designer needs to design good search strategies to optimize the *run-time dynamics* of the algorithm. In general, a good search behavior has the following characteristics: intensify the search on a good region to yield better solutions and diversify when the region is depleted of its potential, e.g. the adaptive/reactive strategies of Reactive Tabu Search (Battiti and Tecchiolli, 1994).

Unfortunately, search strategies are often problem-specific and deriving them is tricky. The effectiveness of search strategies is strongly dependent on the correct timings in which they are applied, which in turn introduce more parameters and rules. Recently, more complex intensifying and diversifying strategies have been proposed in the form of *hybridization*, in which one metaheuristic is hybridized with other metaheuristics and/or with other techniques such as linear programming and branch & bound. While such hybridization can further exploit the beneficial effect of intensification or diversification, it also adds another dimension of complexity in tuning the strategies.

Finding the effective search strategies to enhance the performance of the search algorithm is challenging because the number of possible search strategies can only be limited by one’s own imagination. Moreover, due to several circumstances, a search strategy does not always perform what it is intended to perform as illustrated in various ‘failure modes’ (Watson, 2005). Thus, the need to explore a lot of strategies and then verify its correctness and effectiveness has made this type of tuning problem a tedious task.

2.2 The Need for a Good Solution

Tuning problem is a serious issue. Comments from experts highlight both the importance and the difficulty of addressing this tuning problem:

- “The selection of parameter values that drive heuristics is itself a scientific endeavor, and deserves more attention than it has received in the Operations Research literature…”, (Barr *et al.*, 1995).

- "The design of a good metaheuristic remains an art…", (Osman and Kelly, 1996).
- "For obtaining a fully functioning algorithm, a metaheuristic needs to be configured: typically some modules need to be instantiated (Type-2) and some parameters (Type-1) need to be tuned.", (Birattari, 2004).
- "There is anecdotal evidence that about 10% of the total time dedicated to designing and testing of a new heuristic or metaheuristic is spent on development, and the remaining 90% is consumed (by) fine-tuning (its) parameters", (Adenso-Diaz and Laguna, 2006).
- "…Optimization of an Iterated Local Search may require more than the optimization of the individual components…" and "…There is no a priori single best size for the perturbation. This motivates the possibility of modifying the perturbation strength and adapting it during the run.", Stuetzle in (Glover and Kochenberger, 2003).

Any metaheuristic algorithm designer will face this tuning problem and they must find the solution: metaheuristic that is optimally configured to solve the underlying combinatorial optimization problem under its current context.

Formerly, due to the difficulty of tuning problem, algorithm designers choose to deal with the type-1 and type-2 problems only. Unfortunately, this effort does not guarantee good performance. One may have a good set of components of the metaheuristic algorithm and has all its parameters properly set. But, if the metaheuristic does not exploit adaptive memory and conducting intelligent exploration of the search space, it will often be outperformed by a dynamic, adaptive, self-correcting, and more intelligent counterpart. A simple example has been shown by Reactive-Tabu Search (Battiti and Tecchiolli, 1994), where a good search strategy which is able to adaptively adjusts the tabu tenure can outperform the performance of the original, static Tabu Search, on the set of unknown future instances --- even if the tabu tenure setting of the *static* Tabu Search is the *best* over the set of training instances.

The new classification that we propose put this type-3 tuning problem to be equally important with the other types. Ideally, we believe that to obtain the best solution for the metaheuristic tuning problem, all types of tuning problem must be addressed properly.

In this paper, we propose a new approach for addressing the tuning problem, especially in tuning search strategies.

3. LITERATURE REVIEW

There are several proposals to address the metaheuristic tuning problem. We classify these tuning methods into two major types: Black-Box versus White-Box tuning methods. The details of the classification are shown in the Table X-1.

Table X-1. Black-Box versus White-Box Tuning Methods

	Black-Box Tuning Methods	**White-Box Tuning Methods**
Definition	• Treat metaheuristics as 'black-box'. Usually in form of automated tools that systematically search for the best parameter values or combination of metaheuristic components.	• Open up the 'box' to allow the algorithm designer to inspect the inner-working of the algorithm and to assist in designing a better algorithm. • Require collaboration with human.
Strengths	• Can relieve the burden of addressing type-1 and type-2 tuning problem from human.	• Can address type-1, type-2, and especially type-3 tuning problem. • Allow for possible human creativity, innovation or invention.
Weaknesses	• Do not allow for human creativity, innovation, or invention. • Have difficulty in handling type-3 tuning problem. • Often try too many configurations, thus in tight development time, they can only run/test each configuration in a relatively short period of time.	• Do not relieve the burden of tuning from human. • The human must understand the behavior of the metaheuristic. • Tuning results are inconsistent as different users do tuning differently. • The time required to conduct tuning is also a variable as it depends on the expertise of the user.

3.1 Black-Box Tuning Methods

CALIBRA. (Adenso-Diaz and Laguna, 2006) proposed a tool to automatically calibrate parameter values when given pre-defined ranges. It works by iteratively calling the target algorithm with various set of parameter values and then uses the objective value feedbacks to determine which set of parameter values should be used in the next iteration. CALIBRA uses Taguchi's fractional factorial design to keep the number of parameter values being tried to be within acceptable limit. Iteratively, CALIBRA can narrow down the range of the algorithm parameters until the values converge. After the maximum number of iterations elapsed, CALIBRA will return the best set of parameters found so far. This way, it manages to solve the type-1 tuning problem quite effectively.

CALIBRA has limitations. The current version can only tune up to 5 parameters, the other parameters must be fixed to 'appropriate values'. The need to supply initial range is also problematic when one does not know a good starting range for certain parameters. Furthermore, CALIBRA is not designed to address type-2 and type-3 tuning problem.

F-RACE. (Birattari, 2004) proposed the racing algorithm, a method that was previously known in the machine learning community. The racing algorithm (F-Race), paraphrasing from his work, can be summarized as follows: First, feed F-Race with a (possibly large) set of candidate configurations. F-Race will estimate the expected performance of the candidate configurations in an incremental way and discard the worst ones as soon as sufficient statistical evidence is gathered against them. This allows a better allocation of computing power because rather than wasting time in the evaluation of low-performance configurations; the algorithm focuses on the assessment of the better ones. As a result, more data is gathered concerning the configurations that are deemed yielding better results, and eventually a more informed and sharper selection is performed among them. Finally, the last configuration is declared as the winner (best) configuration. This process is very much analogous with the real life racing.

The number of possible configurations to test can be very large, thus by not trying every possible configuration blindly, F-Race is much better than systematic brute force try-all approach. F-Race is classified as type-1 and type-2 tuning problem solver as it can be used to find good parameter values and proper combination of components of the algorithm simultaneously.

However, F-Race also has several inherent limitations, which arise from the fact that it is a black-box tuning method. Similar to CALIBRA, F-Race is unable to help the algorithm designer to give solution beyond the best configuration found in the set of possible configurations initially supplied to the tuning algorithm. One should also be aware of the ‘combinatorial explosion’ of the number of configurations to be tried, as it will require enormous computation time that may possibly exceeding the maximum allowed development time. Hence, to keep the size of initial set of configurations small, the algorithm designer must intelligently decide which should be included in the set, a process that preferably not be done *blindly*.

3.2 White-Box Tuning Methods

STATISTICAL ANALYSIS. The search space (a.k.a. fitness landscape) of combinatorial optimization problems can be enormously large. Even if one is unlikely to explore the entire search space, one may gather crucial statistical properties of the search space, such as the structure of the search space, the distribution of local optima, the existence of ‘big valleys’, etc. The result of such analysis, if interpreted and reasoned correctly, may yield interesting discoveries that can be exploited to improve the design of the metaheuristic algorithms.

Some of the widely used methods for statistical analysis are Fitness Distance Correlation (FDC) and Run Time Distribution (RTD) analysis. The works by (Fonlupt *et al.*, 1999; Merz, 2000; Hoos and Stuetzle, 2005), are typical works that utilize statistical methods in metaheuristics design.

Statistical methods for metaheuristics design can be used to address all types of tuning problem. However, this process is not straightforward. Knowing the statistical information about the fitness landscape of a combinatorial optimization problem is a necessary but not sufficient condition to design a good metaheuristic for that problem. A significant amount of human effort is still required to reason on the facts found using statistical analysis before a good solution for tuning problem can be produced. In the context of tuning problem, this lengthy process is undesirable due to tight development time. We argue that without a proper computer-aided tool, it is difficult if not impossible to generate the required solution within tight development time, with merely statistical data.

HUMAN-GUIDED SEARCH. As shown in many experiments, human is known to have advantages in visual perception and intelligence over today's computer. Human-guided search tries to utilize these advantages by providing the user with a good visualization and interaction tool to view the problem-specific visualization of the *current solution* (e.g. TSP tours, etc) and to control the search, respectively. In general, human know the ingredients of good solutions of the combinatorial optimization problem, thus human guidance may be able to assist the algorithm to obtain good results quicker.

Research on interactive man-machine optimization can be found in as early as (Michie *et al.*, 1968) and (Krolak *et al.*, 1971). Recently, this line of work is re-surfaced in (Klau *et al.*, 2002).

Human-guided search is an indirect form of white-box tuning method, where one can add the strategies that were adopted when manually guiding the search into the underlying search algorithm. However, guiding the search for a prolonged period of time is tedious in practice as its effectiveness will be limited by the stamina and patience of the human user.

VISUALIZATION OF SEARCH ALGORITHM BEHAVIOR. Rather than visualizing problem-specific information as in human-guided search above, the generic attributes of the search algorithm can also be visualized. By monitoring them, one can gauge the algorithm's performance and can use the information to tune the search algorithm accordingly.

(Kadluczka *et al.*, 2004) proposed a generic visualizer to visualize the search coverage. The authors proposed a mapping for N-dimensional objects to 2-D space which can be displayed on the screen. By plotting the positions of the N-dimensional solutions in 2-D space, one can approximately identify which search space has/has not been explored by the metaheuristic search algorithm. This information can be used as a guidance to tune the algorithm. The limitation of this approach is that the huge gap in size between of the exponential search space and the polynomial screen space renders such visualization inappropriate for larger values of N. Furthermore, the static visualization adopted in this work does not convey the dynamic run-time behavior of metaheuristic search well.

3.3 Remarks

Each tuning method has its own strengths and weaknesses. However, their effectiveness can only be compared relatively --- not only since they are customized to address different types of tuning problems, but also many subjective issues are involved, especially the tuning methods with human intervention. In Table X-2, we provide our subjective view of the differences of each tuning methods with respect to V-MDF as the basis of comparison. In general, most tuning methods have difficulty in handling the type-3 tuning problem. We also observe that most of the works (other than statistical methods) have yet to release their tools for public use (CALIBRA is available on the web[2]).

There are few other works around tuning problem, e.g. agent based approach +CARPS (Monett-Diaz, 2004); self adaptive algorithms (Battiti and Tecchiolli, 1994); metaheuristic to tune other metaheuristic: meta-evolution (Pilat and White, 2002); visualization of 2-variables problem (Syrjakow and Szczerbicka, 1999). All of them belong to either black-box or white-box tuning method depending on whether these methods treat the metaheuristic algorithm being tuned as black-box or not.

Table X-2. Comparison of several factors between existing tuning methods:

Tuning Methods	**A**	**B**	**C**	**D**	**E**	**F**
Type of Method	Black	Black	White	White	White	White
Can address type-1?	Easy	Easy	Hard	Hard	Hard	Average
Can address type-2?	N/A	Easy	Hard	Hard	Hard	Average
Can address type-3?	N/A	N/A	Hard	Average	Average	Easy
Ease of Usage	Easy	Easy	Hard	Average	Average	Average

Legends: **A** (CALIBRA), **B** (F-Race), **C** (Statistical Methods), **D** (Human Guided Search), **E** (Visualization of Search), **F** (V-MDF)

[2] The current version is available at http://coruxa.epsig.uniovi.es/~adenso/file_d.html

4. VISUAL DIAGNOSIS TUNING METHODOLOGY

4.1 Background

The goal of visual diagnosis tuning is to enable the user to address the tuning problem, especially in finding good search strategies quickly, through visual interaction with the search process.

In the past, visualization has been applied for understanding information, e.g. data can be visualized via graphical charts. Good visualization, see (Tufte, 1983, 1990, 1997), conveys information about underlying data or processes and it plays a crucial enabling role in our ability to comprehend large and complex data, to be aware of the situation (*Situation Awareness* theory (Endsley, 2000)). Via such visualization, human can gain insights of the data and possibly, create innovations --- something that is hard to be done by today's computer.

An interaction tool channels the human's idea back to the machine. This cycle of {… – visualization – interaction – visualization – ...} forms the interactive optimization concept as discussed in (Jones, 1996; Scott *et al.*, 2002), and in human guided search that was discussed previously (Michie *et al.*, 1968; Krolak *et al.*, 1971; Klau *et al.*, 2002).

In the context of tuning metaheuristics, if the user is given the proper visualization of the inner-workings of a metaheuristic algorithm, the user may be able to discover interesting properties that are hard for the machine to identify automatically. Such visualization can be used to help answering the 'why' question on the run-time dynamics (i.e. type-3 tuning) of the metaheuristic algorithm, which is the first necessary step to create effective search strategies. As an illustration, suppose that the current results produced by a Tabu Search algorithm are very poor. If presented with the visualization of the inner-workings of Tabu Search plus prior knowledge of the desired Tabu Search behavior, the algorithm designer may become aware of the situation that may be the source of the problem (e.g. the search is trapped in solution cycling) and subsequently, the algorithm designer may find possible treatments to rectify the improper behavior (e.g. increase the tabu tenure).

Visual diagnosis tuning is tied to the metaheuristic algorithm being used and *not* to the combinatorial optimization problem itself, and thus it inherits the generic characteristic of metaheuristic. Hence, such visual diagnosis tuning can be applied to virtually any combinatorial optimization problem as long as a metaheuristic algorithm exists to solve it. For illustrative purpose, our focus in this paper will be for Tabu Search (TS) only.

4.2 {Cause-Action-Outcome} Rules

A ***search trajectory*** is stated as the path taken by the algorithm from the start until the end of the search. Along this trajectory, the search may encounter ***basic events*** (e.g. arrive in local optima, an uphill/downhill move, etc).

We define a more generic term ***incidents*** as the occurrence of a basic event or a sequence/combination of basic events. These incidents can be diagnosed visually to portray the current state of the search trajectory. We define ***positive incidents*** as incidents that shows the search is along a good trajectory (e.g. new best solution found) and ***negative incidents*** as incidents that shows the search is along a bad trajectory (e.g. solution cycling).

In response to negative incidents (or ***cause***), the user might decide to perform a remedial ***action*** – such as adjusting search parameter(s), changing component(s) of the algorithm, or applying intensification/diversification strateg(ies). The hope is that this action will result in a positive, *user-defined* desired incident (or ***outcome***) within a reasonable time. This {cause-action-outcome} sequence is defined and captured as a ***rule***.

To measure the effectiveness of a rule, we compute its ***success score***. The range of success score is (0..1]. In this paper, it is measured by the exponential function: $\mathbf{e^{-\Delta iteration/C}}$ where Δ*iteration* is the number of iterations between the first execution of the action until the observation of the desired outcome. C is a constant and is used to adjust the rate of diminishing success score. We set C to be 30 in this paper. Intuitively, this function dictates that the success score diminishes slowly over time. Observe that the success score = 1 when Δ*iteration* = 0 (the desired outcome is immediately observed) and it tends to 0 when it takes a very long time (or perhaps never) before the desired outcome is observed.

Once a rule is performed, its ***total execution*** counter is incremented by 1, its success score is updated, and the search state is monitored for the next application of the rule. Typically, some action needs several iterations or even re-applied several times before the desired outcome is observable. Thus to avoid excessive re-applications of the action of the same rule, the next check of the search state is done using probability ***(1-success score)***, that is, the action of an effective rule is less likely to be repeated.

The success scores of each rule throughout the search run are then normalized with the total execution to obtain the ***normalized success score (NSS)***, see Figure X-1. This process is repeated over *several* training instances[3] to avoid the danger of 'over-fitting'. If the ***averaged-normalized success score (ANSS)*** of a rule over several training instances is high, the rule is regarded as ***successful*** in bringing the search trajectory into a better one. Otherwise, the rule is regarded as ***less successful*** and the user might

[3] Training instances should have different characteristics, e.g. different problem size.

decide to further adjust his search strategy (action) or to refine the definition of his desired outcome.

By visually diagnosing the transformation from the cause incident to the outcome incident and monitoring the averaged-normalized success score, the user can determine the effectiveness of his search strategy.

For example, the high averaged-normalized success score of:

{Non_Improving – Greedy_Random_Restart – At_Good_Region}

signifies the potential effectiveness of this *greedy* random restart strategy to steer the search from bad region to area with good quality solutions; whereas the almost zero averaged-normalized success score of:

{Solution_Cycling – Decrease_Tabu_Tenure – No_Solution_Cycling}

shows the ineffectiveness of decreasing the tabu tenure during solution cycling, and:

{Solution_Cycling – 'Magic_Strategy' – Reach_Optimal_Solution}

illustrates an almost impossible scenario where only a 'magic strategy' can achieve the overly optimistic desired outcome.

We argue in this work that high averaged-normalized success score is a strong measure of the effectiveness of a remedial action, as high score implies that the action frequently steers the search trajectory from negative incidents to desired (positive) outcomes, at least over the several training instances. This argument carries weight if we assume further that future instances have similar characteristics with training instances.

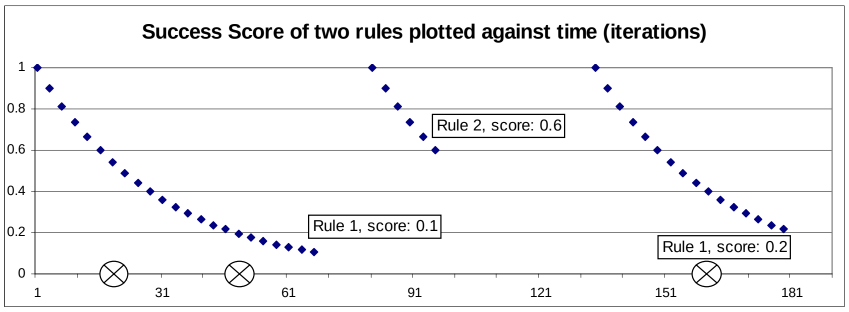

Figure X-1. This is an example of the success scores of the execution of two rules. The ⊗ sign along the X-axis marks re-applications of the action of the rule before the search manages to arrive at the desired outcome. Here, Rule 1 is executed twice. The scores of 0.1 and 0.2, which is normalized over two executions to obtain NSS of 0.15, imply that either the strategy or the formulation of desired outcome adopted in Rule 1 has problem; whereas the high NSS of 0.6 of Rule 2 implies the potential effectiveness of the strategy used in Rule 2. These rules will then be applied to other training instances to obtain ANSS.

5. VISUALIZER FOR MDF (V-MDF)

V-MDF is a white-box tuning tool that utilizes the visual diagnosis tuning methodology to address the type-3 tuning problem. In this section, we present the two main components of V-MDF: ***Distance Radar*** and ***Rule-Base***, followed by a discussion on how V-MDF is used for visual diagnosis tuning.

5.1 Distance Radar

The Distance Radar is the underlying graphical user interface for visualizing incidents in the search trajectory. The function of the Distance Radar in this paper is to display incidents that occur along a Tabu Search trajectory. These incidents either indicate the necessity for a remedial action or to display the outcome of an applied strategy. From these incidents, the user can derive rules in form of {cause-action-outcome} discussed above.

Essentially, Distance Radar graphically plots generic properties of ***distance***[4]***, fitness (objective value), and recency*** information of the elite solutions with respect to the *current solution*. In the trajectory based search, the current solution can be seen as the '***current position***' in the search space and the elite solutions, which were found and recorded along the search trajectory traversed so far, can be seen as the signposts or ***anchor points*** in the search space. By measuring the distance of current solution (current position) to these 'anchor points', coupled with the other two generic properties: fitness and recency information, one can gain information of the relative movement of the search along its trajectory with respect to these 'anchor points'. This new search trajectory tracking concept enables the user to visualize the previously infeasible search trajectory visualization. This is because the size of the set of recorded elite solutions/anchor points is fixed and much smaller than the exponential size of the search space.

The Distance Radar consists of dual 2D graphs: **Radar A** (with **Recency graph**) and **Radar B** (with **Fitness graph**). Each of the radar is used to exhibit distance information from different perspective. In both radars, the X-axes represent the anchor points and Y-axes show the distance between current solution with each of the anchor point. The Y-axes is drawn in logarithmic scale to emphasize the importance of anchor points within short distances with respect to the current solution. Points in the radars are connected with lines to help the user in diagnosing the trend.

[4] Discussions about various distance functions can be found in (Sevaux and Soerensen, 2005; Ronald, 1997,1998; Fonlupt *et al.*, 1997), etc. For example, the user can use 'bond distance' and 'hamming distance' to measure the distance of two Traveling Salesman Problem (TSP) and two Military Transport Planning (MTP) solutions, respectively.

Radar A displays the sorted anchor points by their fitness values. The recency values of these anchor points are plotted in the complementary **Recency graph**. Radar A displays only a visually manageable number of anchor points (a small but adjustable fraction with respect to the problem size) and any better elite solution found will replace the poorest recorded anchor point. The effect of Radar A is to approximate the 'goodness' of the region currently being searched. Generally, if Radar A shows a trend that is gradually moving upward (distance to current solution increases) from some anchor points, it indicates that the search is diversifying from the region of these anchor points. On the other hand, if the trend is moving downward, the search is intensifying onto region near these anchor points.

Radar B displays the sorted anchor points by their recency. The fitness of these anchor points is plotted in the complementary **Fitness graph**. Typically the number of recent solutions being recorded is set to be the same as the tabu tenure. Radar B can be seen as a long-term memory mechanism that complements the tabu list (short term memory). As cycling usually occurs around these recently visited solutions (especially local optima), Radar B can detect cycling in them quickly.

All the graphs: Radar A, Recency graph, Radar B, and Fitness graph complements each other to help a user in detecting various incidents. Figure X-2 illustrates some incidents observeable using these graphs, e.g. *solution cycling, plateau effect, non-improving*, etc.

Figure X-3 illustrates an example of how the observation of the incidents via Distance Radar can assist the selection of a remedial action. In this example, three elite solutions/anchor points have been found along the search trajectory of a minimizing problem and recorded as Local Optima 1, 2, 3. Now, suppose from the 3rd local optima to current solution, the search experienced a series of non-improving solutions (drawn as dotted lines from Local Optima 3 to current solution). The situation (***cause***) triggered the need for a remedial action. At this point, the algorithm designer may attempt to improve the search by applying a search strategy Z (***action***). Let one of either Solution X or Y be the solutions (***outcome***) reached after applying the strategy Z.

For Solution X, Radar A shows that the search is heading towards the current best local optima solution and Radar B shows that the nearest local optima solution is the one that is 2nd most recently found. Both radars have shown the algorithm designer that after applying strategy Z, the search is heading towards good recently found local optima. Hence, if strategy Z is intended to perform intensification, the observation from the Radar plots shows that it is indeed on the right track; otherwise it is considered as ineffective (as moving towards Solution X is not its intended purpose).

For Solution Y, Radar A and B shows an upward moving horizontal line. This indicates that the current solution is moving away from all known local optima solutions, which is the ‘correct’ outcome if the purpose of strategy Z is to conduct diversification.

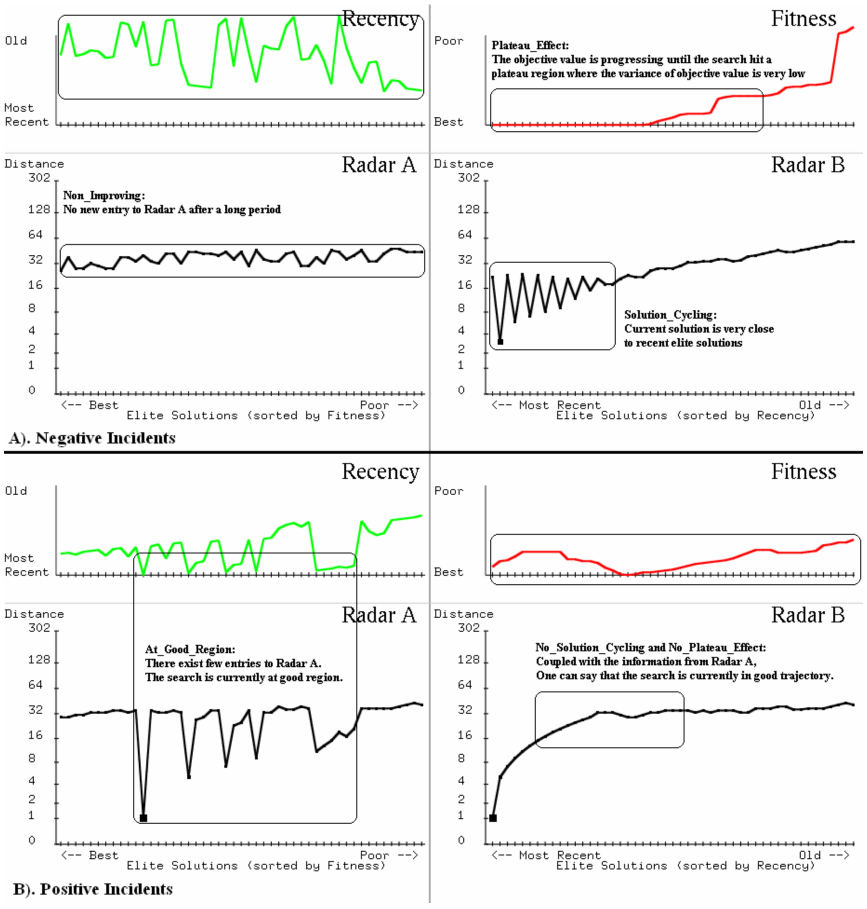

Figure X-2. Examples and interpretations of several incidents: negative (above) and positive (below).

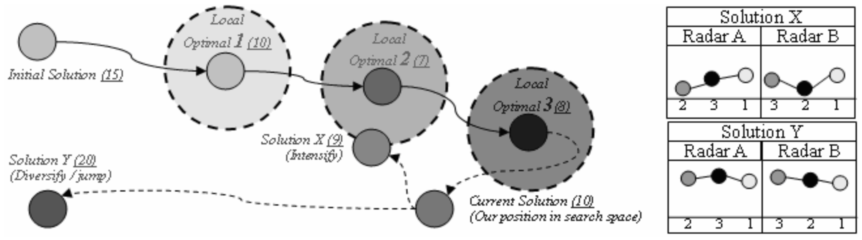

Figure X-3. This is a visualization of the search trajectory of a minimizing problem. Without the aid of Distance Radar, it is hard to see the search behavior. On the other hand, one can understand the run-time dynamics of the Tabu Search algorithm by observing the plots shown in Distance Radar (moving to X is intensification to region around solution 2 and moving to Y is diversification).

5.2 Rule-Base

The {cause-action-outcome} rules that are derived while observing the incidents using Distance Radar are stored in a repository called the **Rule-Base (RB)**, which maintains the normalized success scores of the rules.

Upon completing visual diagnosis tuning, the user may examine the RB to decide whether to discard statistically inferior rules. The rules that survive eventually will form the basis for the solution of the tuning problem, in the sense that these rules can be either left as search strategies (triggered as needed/type-3) or merged into the metaheuristic algorithm (by modifying the parameters/type-1 or components of the algorithm/type-2).

The {cause-action-outcome} rules are implemented by the event-driven mechanism of MDF (Lau *et al.*, 2004b, 2006), as follows.

First, the user implements a V-MDF’s EVENT class that describes an incident and links it with the desired HANDLER class. The user can refine the implementation of these EVENT classes to adjust the accuracy of the sensing of those incidents. The examples of the pseudo-code of V-MDF EVENT classes that describe two negative incidents (which were shown previously in Figure X-2) are listed in Table X-3a.

Next, the user needs to define a remedial action: the necessary steps required to alter the search trajectory, in form of the HANDLER class. When V-MDF senses an EVENT (cause) for the first time, it will trigger the associated HANDLER (action), register the ID of the desired outcome in the V-MDF desired_event table, and increment the total execution of the associated rule. However, subsequent re-applications of the action of the same rule will be done using (1-success score) probability. The example of the pseudo-code of V-MDF HANDLER classes is shown in Table X-3b.

Finally, V-MDF will automatically check the search state after the execution of the HANDLERs with another EVENT (desired outcome). These events have no associated HANDLERs. If this EVENT is expected to occur (by checking the IDs listed in desired_event table), then the success score of the associated rule is computed using the formula explained in Section 4.2. The desired outcomes that occur after a long time will obtain a very low score. The example of the pseudo-code of V-MDF EVENT classes for checking desired outcome is shown in Table X-3c.

Table X-3. Examples of cause (EVENT), action (HANDLER), & outcome (EVENT) in ***pseudo-code***.

Non_Improving : Event {
return true if there is no new entry to Radar A after a long period, return false otherwise;
}
Set Handler: 'Greedy_Random_Restart'
Add total execution of this rule by 1.
	Solution_Cycling : Event {
	return true if the distance to one or more recent elite solutions in Radar B is short, return false otherwise;
	}
	Set Handler: 'Increase_Tabu_Tenure'
	Add total execution of this rule by 1.
		A. CAUSE

Greedy _Random_Restart : Handler {
pick TS current best solution, perturb it in greedy fashion, and set TS to resume from the newly created solution;
}
Add 'At_Good_Region'
in desired_event table.
	Increase_Tabu_Tenure : Handler {
	get TS current tabu tenure,
	increase it a bit,
	set current tabu tenure to the new value;
	}
	Add 'No_Solution_Cycling'
	in desired_event table
		B. ACTION

At_Good_Region : Event {
return true if the fitness difference between the current and best found local optima is low; return false otherwise;
}
Add the success score of the associated rule if this event's ID is found in desired_event table.
	No_Solution_Cycling : Event {
	return true if the distance to all recent elite solutions in Radar B are far enough, return false otherwise;
	}
	Add the success score of the associated rule if this event's ID is found in desired_event table.
		C. OUTCOME

Table X-4. Overall Workflow of V-MDF

A. Implementation Phase
- Implement the metaheuristic algorithm in **MDF framework** (Lau *et al.*, 2004b, 2006)

B. Visual Diagnosis Tuning Phase
- Using V-MDF's **Distance Radar**, diagnose the run-time dynamics of the metaheuristic algorithm when applied to several representative **training instances**.
- For each negative incident that requires an action,
  Write the appropriate {cause (EVENT), action (HANDLER), outcome (EVENT)} rule.
  The success score and total execution of rules will be monitored by **Rule-Base**.
- Human can further diagnose (visually), add new rules, modify or delete existing rules.

C. Rules Selection Phase
- Turn off V-MDF's Distance Radar.
- User can discard rules with low **ANSS** success score. (e.g. instance-specific rules).
- Surviving rules in Rule-Base form the elements of the final metaheuristic algorithm.
  1. Leave the rules as search strategies, or
  2. Merge the rules into the algorithm (i.e. the rules become native to the algorithm).

D. Testing Phase
- Test the metaheuristic algorithm with **good rules** to the **whole test instances**.

5.3 Putting It All Together

The workflow for implementing and tuning a metaheuristic to solve a combinatorial optimization problem using V-MDF is outlined in Table X-4.

6. EXPERIMENTAL RESULTS

In this section, we report the experimental results. The real-life and artificially generated test instances, plus several executables of V-MDF are available at http://www.comp.nus.edu.sg/~stevenha/v-mdf.

6.1 Test Problem: Military Transport Planning (MTP)

We applied V-MDF to tune a Tabu Search implementation for solving an NP-hard combinatorial optimization problem: Military Transport Planning (MTP) which was defined in (Lau *et al.*, 2004a):

> Given service level **q** and a set of **n** requests from military units in tuple: {number_of_vehicle_required, start_time, end_time}, choose **q** out of **n** requests such that the total number of vehicles required to serve all **q** requests is minimized. number_of_vehicle_required ≥ 1 and [start_time .. end_time] lies within the range of a predetermined planning horizon.

Besides experimenting with several real life instances of this problem, we artificially created larger test instances with known optimal values as follows. First, create ***x*** random requests and then compute in polynomial time, the minimum number of vehicles ***z*** that is required to satisfy *all* |***x***| requests. Finally, insert ***y*** pairs of dummy requests such that ***q*** = |***x***| + |***y***| and ***n*** = |***x***| + ***2*** * |***y***|. In this pair of dummy requests ***(y,y')***, every attempt to include ***y*** will not increase ***z*** while for ***y'***, it will always increase the number of vehicles required. The optimal solution is only one: first ***x*** requests plus all ***y*** requests, ignoring the entire ***y'*** requests. The value ***z*** will be the optimal value for this artificial test instance.

6.2 Experimental Methodology

The purpose of our experiment is to demonstrate the capability of V-MDF in dealing with the tuning problem that arises during the implementation of a Tabu Search algorithm for MTP. All experiments are conducted using an Athlon XP 2500+ machine with the following specifications: 1.8 GHz, 512 MB RAM, Windows XP. All codes are developed using VC++ .NET 2003.

The experimental methodology is as follows:

1. Prepare a set of real-life and artificially generated test instances.
2. Start with a quick-and-dirty implementation (see Table X-5).
3. Record the results for all test instances. Tabu Search runs for 1000 iterations for each test instance (see Table X-7, column 'Before').
4. Tune Tabu Search algorithm with V-MDF using two training instances (T4 and T7). The tuning time taken for the first author to conduct the tuning for the first attempt is approximately 10 man hours.
5. Verify rules in Rule-Base (see Table X-6) in terms of their effectiveness.
6. Record the results of the tuned algorithm for all test instances again, using the same 1000 iterations limit (see Table X-7, column 'After').
7. Compare the results.

6.3 Initial Results

Without proper insights on what happens within the search itself, one can only guess which part of the algorithm that needs to be tuned. The only observable fact without using the tool like V-MDF is the trend that the performance of this Tabu Search implementation deteriorates when problem size gets larger (See Table X-7, column 'Before'). With V-MDF and its Distance Radar, one can detect the possible problems and tune the Tabu Search accordingly.

Table X-5. Quick-and-dirty TS implementation for MTP using MDF software framework

Component	Remark
Solution	The solution representation is simply a bit string *b* of size *n*. *b*[*i*] = 0 when request *i* is not satisfied and 1 otherwise.
Initial Solution	Randomly select *q* requests (the seed is fixed for all the experiments)
Local Move and Neighborhood	Bit-flip move that will transform solution *b* to *b'* with 1 bit changed. ($d_{hamming\_distance}(b,b') = 1$). Thus we have a maximum of O(n) possible neighbors per iteration. Infeasible neighbors are penalized by adding a constant penalty of 1000.
Objective Function	For each satisfied request, add its vehicle requirement to the histogram. The objective value is the maximum value in the resulting histogram.
Tabu List	Same bit flip move can't be applied for the next *tabu_tenure* iterations.
Tabu Tenure	*tabu_tenure* is initially set as 0.1 * *n*.
Search Strategies	None.

6.4 Tuning Phase

The first problem visually observed is the so-called 'Plateau_Effect'. This phenomenon can be easily explained: The objective values of MTP solutions are discrete and their range is small, thus logically, there will be many MTP solutions that have similar objective value. 'Plateau_Effect' can severely reduce the effectiveness of neighborhood based search. We try several methods and arrive with a penalty function that penalizes *very infeasible* solutions more than *slightly infeasible* solutions and also penalizes solutions that are too far from good solutions found so far. These two modifications help reducing the plateau effect.

The second visual observation reveals ‘Solution_Cycling’ incidents. We apply V-MDF to observe the behavior of the algorithm while we adjust the tabu tenure, emulating Reactive-TS strategy. We then add a greedy random restart strategy where we perturb the best found solution by randomly pick requests with small number_of_vehicle_required. This acts as a diversifier to enhance the search when it encounters ‘Non_Improving’ incidents.

All the rules found during the tuning process and their success rates against the training instance are listed in Rule-Base (see Table X-6). Based on the statistical data, we discard the ineffective rules; merge some of the effective rules into the final algorithm; while the remaining rules are left as search strategies. The results of the algorithm are recorded in Table X-7.

6.5 Results after Tuning

We observe in Table X-7 that the result improves substantially compared to the initial results after a relatively short tuning phase. We like to point out that the result per se does not matter much, but rather it is the manner that V-MDF has helped the algorithm designer to identify negative incidents in a timely fashion that is essentially helpful to the tuning of the algorithm. This simple experiment has shown that by understanding the problems encountered by the search algorithm on-the-fly, albeit imperfectly, one can provide better remedies for such problems much faster, compared to blind trial-and-error.

7. CONCLUSIONS AND FUTURE WORKS

In this paper, we studied the issue of tuning metaheuristics through visualization. An extensive review of the existing tuning methods reveals that works in the literature are scarce in handling the type-3 tuning problem. We proposed a new visual diagnosis tuning methodology to address this tuning problem. We presented a generic visualizer tool V-MDF to support this methodology. V-MDF is currently designed for tuning Tabu Search strategies.

Table X-6. This is the content of the Rule-Base after conducting visual diagnosis tuning using T4 and T7. Observe the column ANSS of a rule over *multiple* (two) training instances. The closer the value to 1.0, the better that rule is. Statistically inferior rules are discarded; good rules are either merged into the final metaheuristic algorithm or left as search strategies.

Cause	**Action**	**Desired Outcome**	**NSS**		**ANSS**
			Over T4	**Over T7**	
Effective rules, merged into the original algorithm					
Plateau_Effect	Apply_Penalty_Function	No_Plateau_Effect	-	-	-
Effective rules, left as search strategies					
Solution_Cycling	Increase_Tabu_Tenure	No_Solution_Cycling	4.4/5: <u>0.87</u>	8.1/10: <u>0.81</u>	**0.84**
Passive_Search	Decrease_Tabu_Tenure	Aggressive_Search	2.6/4: <u>0.66</u>	3.9/6: <u>0.66</u>	**0.66**
Non_Improving	Greedy_Random_Restart	At_Good_Region	2.9/4: <u>0.71</u>	5.9/7: <u>0.84</u>	**0.77**
Discarded rules (purposely listed here as illustration)					
Solution_Cycling	Decrease_Tabu_Tenure	No_Solution_Cycling	0.76/2: <u>0.38</u>	6.6/11: <u>0.60</u>	**0.49**

Table X-7. Table of experimental results: before and after tuning. Test instances are divided into two categories and ordered by problem size. T4 and T7 are used as the training instance (shaded) and should not be considered for the evaluation of the final algorithm performance. Observe the improvement of the tuned over the non-tuned algorithm as well as the gap to optimal (for artificially generated test instances).

MTP Test Instances				Vehicles Required			Gap to Optimal	
				Before	After	Optimal	Before	After
Real-life test instances								
T1	n: 39	q: 31	(80%)	6	5	-	-	-
T2	n: 249	q: 186	(75%)	61	35	-	-	-
T3	n: 283	q: 240	(85%)	84	84	-	-	-
T4	n: 302	q: 250	(83%)	277	140	-	-	-
Randomly generated test instances with known optimal								
T5	n: 50	q: 40	(80%)	33	18	16	17 (106%)	2 (13%)
T6	n: 100	q: 85	(85%)	37	37	35	2 (06%)	2 (06%)
T7	n: 200	q: 180	(90%)	54	33	31	23 (74%)	2 (07%)
T8	n: 300	q: 250	(83%)	45	32	24	21 (88%)	8 (33%)
T9	n: 400	q: 300	(75%)	147	87	75	72 (96%)	12 (16%)

Our experience shows that V-MDF is effective in helping the user discover and rectifying negative incidents through proper remedial actions. We believe it is possible to develop a better way for visualizing Tabu Search or other metaheuristic search strategies via statistical methods such as fitness distance correlation plots. We hope to enhance V-MDF by providing decision support for the user to detect negative incidents, to choose better remedial actions, and to measure the performance of the rules. Collaboration between V-MDF and automated methods is also another possible future work. Finally, we see the prospect of using V-MDF as a research tool to invent search strategies not yet known at present.

The progress in metaheuristics research is rapid, but the end-users still require down-to-earth, ready-to-use tools for tuning their metaheuristic algorithms. Currently, research involving metaheuristics tuning problem is still preliminary and there are not many good tools available for public usage. However, we anticipate that several of the tuning methods that are theoretical concepts today will become widely used tools for metaheuristic algorithm design in the near future.

POSTSCRIPT

We have since expanded Distance Radar into a more generic, off-line visualization tool called *Viz* (see Halim *et al.*, 2006).

ACKNOWLEDGEMENTS

The authors acknowledge inputs from Wan Wee Chong for the earlier version of this work, Roland Yap, Mauro Birattari (IRIDIA, Belgium) especially for the discussion of tuning problem, and the anonymous referees.

REFERENCES

Adenso-Diaz, B., and Laguna, M., 2006, Fine-tuning of Algorithms Using Fractional Experimental Designs and Local Search, *Operations Research* **54**(1): 99-114.

Barr, R.S., Golden, B.L., Kelly, J.P., Resende, M.G., and Stewart, W.R., 1995, Designing and Reporting on Computational Experiments with Heuristic Methods, *Journal of Heuristics* **1**:9-32.

Battiti, R., and Tecchiolli, G., 1994, The Reactive Tabu Search, *ORSA Journal on Computing* **6**(2): 126-140.

Birattari, M., 2004, The Problem of Tuning Metaheuristics as seen from a machine learning perspective, PhD Thesis. University Libre de Bruxelles.

Charon, I., and Hudry, O., 1995, Mixing Different Components of Metaheuristics, In *Meta-Heuristics: Theory and Applications*, Osman, I.H. and Kelly, J.P., ed.: Kluwer Academic Press: 589-603.

Endsley, M.R., 2000, Theoretical Underpinnings of Situation Awareness: A Critical Review, in: *Situation Awareness Analysis and Measurement*, Endsley and Garland, ed: Lawrence Erlbaum Associates, Mahwah, NJ.

Fonlupt, C., Robilliard, D., Preux, P., and Talbi, E., 1999, Fitness Landscapes and Performance of Meta-heuristics, in: *Meta-Heuristics - Advances and Trends in Local Search Paradigms for Optimization*, Voss, S., Martello, S., Osman, I.H., Roucairol, C., ed.: Kluwer Academic Press, 18: 255-266.

Glover, F. and Kochenberger, G., 2003, *Handbook of Metaheuristics*, Kluwer Academic Publishers.

Halim, S., Yap, R., and Lau, H.C., 2006, Viz: A Visual Analysis Suite for Explaining Local Search Behavior, To appear in 19th Annual ACM Symposium on User Interface Software and Technology (UIST'06).

Hoos, H.H. and Stuetzle, T., 2005, *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann.

Jones, T., and Forrest, S., 1995, Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms, In Proceedings of 6th International Conference on Genetic Algorithms (ICGA'95): 184-192.

Jones, C.V., 1996, *Visualization and Optimization*, Kluwer Academic Publishers.

Kadluczka, M., Nelson, P.C., and Tirpak, T.M., 2004, N-to-2-Space Mapping for Visualization of Search Algorithm Performance, In Proceedings of 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04): 508-513.

Klau, G.W., Lesh, N., Marks, J., and Mitzenmacher, M., 2002, Human-Guided Tabu Search, In Proceedings of 18th National Conference on Artificial Intelligence (AAAI'02): 41-47.

Krolak, P., Felts, W., and Marble, G., 1971, A Man-Machine Approach Toward Solving The Traveling Salesman Problem, Communications of the ACM **14**(5): 327-334.

Lau, H.C., Ng, K.M., and Wu, X., 2004a, Transport Logistics Planning with Service-Level Constraints, In Proceedings of 19th National Conference on Artificial Intelligence (AAAI'04): 519-524.

Lau, H.C., Wan, W.C., Lim, M.K., and Halim, S., 2004b, A Development Framework for Rapid Meta-Heuristics Hybridization, In Proceedings of International Computer Software and Applications Conference (COMPSAC'04): 362-367.

Lau, H.C., Wan, W.C., Halim, S., and Toh, K. 2006, A Software Framework for Fast Prototyping of Meta-heuristics Hybridization, To appear in Special Issue of *International Transactions in Operational Research* (ITOR).

Merz, P., 2000, Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies, PhD Thesis. University of Siegen, Germany.

Michie, D., Fleming, J.G., and Oldfield, J.V., 1968, A Comparison or Heuristic, Interactive, and Unaided Methods of Solving a Shortest-Route Problem. In *Machine Intelligence*, Michie, D., ed: American Elsevier Publishing Co., New York: 245-255.

Monett-Diaz, D., 2004, Agent-Based Configuration of Metaheuristic Algorithms, PhD Thesis. Humboldt University of Berlin.

Osman, I.H. and Kelly, J.P., 1996, *Meta-heuristics – The Theory and Applications*, Kluwer Academic Publishers.

Pilat, M.L. and White, T., 2002, Using Genetic Algorithms to optimize ACS-TSP. In Proceedings of the 3rd International Workshop on Ant Algorithms (ANTS 2002):282-287.

Ronald, S., 1997, Distance functions for order-based encodings, In Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97): 43-48.

Ronald, S., 1998, More distance functions for order-based encodings, In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation (ICEC'98): 558-563.

Scott, S.D., Lesh, N., and Klau, G.W., 2002, Investigating Human-Computer Optimization, In Proceedings of Conference on Human Factors in Computing Systems (CHI'02): 155-162.

Sevaux, M., and Soerensen, K., 2005, Permutation distance measures for memetic algorithms with population management, In Proceedings of 6th Metaheuristics International Conference (MIC'05).

Syrjakow, M. and Szczerbicka, H., 1999, Java-based animation of probabilistic search algorithms, In Proceedings of International Conference on Web-based Modeling and Simulation: 182-187

Tufte, E., 1983, *The Visual Display of Quantitative Information*, Graphic Press.

Tufte, E., 1990, *Envisioning Information*, Graphic Press.

Tufte, E., 1997, *Visual Explanations*, Graphic Press.

Watson, J.P., 2005, On Metaheuristics "Failure Modes": A Case Study in Tabu Search for Job-Shop Scheduling, In Proceedings of 6th Metaheuristics International Conference (MIC'05).