On Metaheuristic Algorithms for Combinatorial Optimization Problems

Mutsunori YAGIURA and Toshihide IBARAKI

Graduate School of Informatics,
Kyoto University, Kyoto 606-8501, Japan
{yagiura, ibaraki}@i.kyoto-u.ac.jp

Abstract: Metaheuristic algorithms are widely recognized as one of the most practical approaches for combinatorial optimization problems. Among representative metaheuristics are genetic algorithm, simulated annealing, tabu search and so on. In this paper, we explain essential ideas used in such metaheuristic algorithms within a generalized framework of local search. We then conduct numerical experiment of metaheuristic algorithms using rather simple implementations, to observe general tendencies of their performance. From these results, we propose a few recommendations about the use of metaheuristics as simple optimization tools. We also mention some advanced techniques to enhance the ability of metaheuristics. Finally, we summarize some theoretical results on metaheuristic algorithms.

Keywords: combinatorial optimization problem, approximate algorithm, metaheuristics, local search, genetic algorithm, simulated annealing, tabu search.

1 Introduction

Optimization problems are generally formulated as follows:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subjectto} & x \in F. \end{array} \tag{1}$$

We call f the *objective function*, F the *feasible region* that satisfies all the given constraints, and a solution $x \in F$ a *feasible solution*. If F has combinatorial features, then problem (1) is called a *combinatorial optimization problem*.

There are numerous combinatorial optimization problems, for which computing exact optimal solutions is computationally intractable; e.g., those known as NP-hard [51]. However, in practice, we are usually satisfied with good solutions, and in this sense, *approximate* (or *heuristic*) *algorithms* are very important. In addition to common heuristic methods such as *greedy method* and *local search* (abbreviated as LS; also called as *hill climbing method* or *neighborhood search*), there is a recent trend of *metaheuristics* which combine such heuristic tools in more sophisticated frameworks.

Those metaheuristics include *random multi-start local search* (abbreviated as MLS) [90, 101, 102, 126, 136, 135], *genetic algorithm* (abbreviated as GA; also called as *evolutionary computation*[1]) [34, 68, 76, 94, 115, 149], *simulated annealing* (abbreviated as SA) [1, 3, 24, 86, 92], *tabu search* (abbreviated as TS) [55, 59, 62, 64, 75] and so on. Among variants of these are *genetic local search* (abbreviated as GLS) [17, 18, 49, 82, 95, 114, 146, 151], which incorporates LS into GA, *greedy randomized adaptive search procedure* (abbreviated as GRASP) [43, 45, 46, 47, 97, 99, 137], which uses randomized greedy methods to generate initial solutions for LS, *iterated local search* (abbreviated as ILS) [85, 110], which uses good solutions found in the previous search to generate initial solutions for LS, *variable neighborhood search* (abbreviated as VNS) [19, 20, 73, 112], which is a variant of ILS, a simplified variant of SA called *threshold accepting* (abbreviated as TA) [41, 100], a variant of TA called *great deluge algorithm* (abbreviated GDA) [40, 144], a simplified variant of TS called *guided local search* [91, 150, 153, 154] and so on.

These metaheuristic algorithms are based on the iteration of the following two steps:

1) search new solutions on the basis of the previous history, and
2) evaluate the solutions generated in Step 1 and extract necessary information for the future search.

Therefore, metaheuristics can be considered as the collection of ideas of how to use the search history to generate new solutions and how to extract the necessary information from the generated solutions. In this paper, we survey various metaheuristic algorithms and give a systematic view by considering them as generalizations of MLS. Such systematic treatment of metaheuristics is quite important, as various metaheuristic ideas have been combined to make the search more effective in recent years. It is also hoped that this sort of survey would motivate new metaheuristic ideas inspired by flexible thought.

One of the attractive features of metaheuristics is in its simplicity and robustness. They can be developed even if deep mathematical properties of the problem domain are not at hand, and still can provide reasonably good solutions, much better than those obtainable by simple heuristics. Moreover, in many cases, their performance will not degrade much even if the problem structures and/or parameters of algorithms are changed.

We pursue this direction in Section 4, by implementing various metaheuristic algorithms and comparing their performance (a part of the results is also found in [156, 157, 159]). The objective of this experiment is not to propose the most powerful algorithm but to compare general tendencies of various algorithms. The emphasis is placed not to make each ingredient of the tested metaheuristics too sophisticated, and to avoid detailed tuning of the program parameters involved therein, so that practi-

[1]To be precise, the terminology "evolutionary computation" is more general than GA.

tioners can easily test the proposed framework to solve their problems of applications. As concrete problems to test, we solve the *single machine scheduling problem* (abbreviated as SMP), and the *maximum satisfiability problem* (abbreviated as MAXSAT), both of which are known to be NP-hard. SMP asks to determine an optimal sequence of n jobs that minimizes a cost function defined for jobs. For MAXSAT, we are given n 0-1 variables, m clauses defined on the variables and their negations, and weights of clauses. Then we are asked to determine a 0-1 assignment that maximizes the sum of the weights of satisfied clauses. Based on the computational results on these problems, taking into consideration the manpower required for the programming (the evaluation of the labor is subjective to some extent), we give a guideline on the use of metaheuristic algorithms as simple optimization tools.

Another attractive feature of metaheuristics is in its flexibility. They can incorporate various ideas into the algorithm to make it more effective. As metaheuristic algorithms usually have many parameters to control the search, which have to be carefully tuned to get good performance, adaptive control mechanism of such parameter values is also important. In Section 5, we introduce various advanced techniques of this type to improve the performance of metaheuristic algorithms.

We also briefly summarize theoretical results on metaheuristics in Section 6. Among them are the asymptotic convergence of the simulated annealing (SA), the complexity class PLS (polynomial-time local search), which was introduced to analyze the computational complexity of LS theoretically, and the approximation ratio of LS.

Here we give the definitions of problems SMP and MAXSAT, which are used in the computational experiments in Section 4. (For details on the generation of instances, see Sections 4.1 and 4.4.) The single machine scheduling problem (SMP) is one of the representative scheduling problems, to which much effort has been devoted [12, 81]. SMP asks to determine an optimal sequence of n jobs in $V = \{1, \ldots, n\}$, which are processed on a single machine. The machine can handle only one job at a time, while permitting no preemption or idle time between jobs. Thus a schedule is determined by a sequence σ of n jobs. A sequence σ: $\{1, \ldots, n\} \to V$ is a one-to-one mapping such that $\sigma(k) = i$ means that job i is the k-th job processed on the machine. Each job i becomes available at time 0, requires integer processing time p_i and incurs cost $g_i(c_i)$ if completed at time c_i, where c_i is uniquely determined by σ as $c_i = \sum_{k=1}^{\sigma^{-1}(i)} p_{\sigma(k)}$. Then the problem is formulated as follows.

Single machine scheduling problem (SMP)

Input: The set of n jobs $V = \{1, \ldots, n\}$, and for each job $i \in V$, a processing time p_i and a cost function g_i of the completion time.

Output: A sequence σ that minimizes $cost(\sigma) = \sum_{i \in V} g_i(c_i)$.

The maximum satisfiability problem (MAXSAT) is a representative combinatorial

optimization problem, which has been extensively studied [72, 142, 143]. Variables $y_1, \ldots, y_n$ and their negations $\bar{y}_1, \ldots, \bar{y}_n$ are called *literals*, and the logical sum of a subset of given literals (e.g., $C_i = y_1 \vee \bar{y}_3 \vee y_4$) is called a *clause*. For MAXSAT, we are given n 0-1 variables $y_1, \ldots, y_n$, m clauses $C_1, \ldots, C_m$ and weights $w_1, \ldots, w_m \in Z_+$ of clauses (Z_+ is the set of nonnegative integers). Let $v = (v_1, \ldots, v_n) \in \{0,1\}^n$ be a 0-1 assignment of $y = (y_1, \ldots, y_n)$, i.e., $v_j = 1$ implies $y_j = 1$ and $\bar{y}_j = 0$, and $v_j = 0$ implies $y_j = 0$ and $\bar{y}_j = 1$. We denote by $C_i(v)$ the value of clause C_i for a 0-1 assignment v. Then the problem is formulated as follows.

Maximum satisfiability problem (MAXSAT)
Input: Clauses $C_1, \ldots, C_m$ of n variables and weights $w_1, \ldots, w_m$ of clauses.
Output: A 0-1 assignment $v \in \{0,1\}^n$ that maximizes $gain(v) = \sum_{i=1}^{m} w_i C_i(v)$.

The *satisfiability problem* (abbreviated as SAT) is the decision problem asking whether a 0-1 assignment v satisfying $\sum_{i=1}^{m} C_i(v) = m$ exists or not. SAT is well-known as the first NP-complete problem [51].

Here we also give the definitions of two problems, *graph coloring* and *traveling salesman*, as we will use them in the subsequent sections.

Graph coloring problem (GCP)
Input: An undirected graph with n vertices.
Output: A coloring of the vertices with the minimum number of colors, in which the end vertices of each edge receive different colors.

Traveling salesman problem (TSP)
Input: The set of n cities and a distance d_{ij} for each pair i and j of cities ($i, j = 1, \ldots, n$).
Output: A *tour* of the minimum total distance, where a tour is a closed path that visits each city exactly once and returns to the first city.

These are also representative combinatorial optimization problems, which are known to be NP-hard [51]. In this paper, we assume that the distance matrix (d_{ij}) is symmetric, i.e., $d_{ij} = d_{ji}$ holds for all pairs of i and j. Wile a solution of TSP is usually represented by a sequence of cities, it is also represented by the set of pairs of cities adjacent in the tour, which are called tour edges (or simply edges).

2 Representative metaheuristic algorithms

In this section, we first discuss the search space, which is quite important in designing metaheuristic algorithms. Then we introduce two basic approaches to approximate algorithms, greedy method and LS, and four representative metaheuristic algorithms,

MLS, GA, SA and TS. Their variants will be explained in the next section. For more details of metaheuristics, see for example [2, 78, 79, 96, 121, 122, 123, 128, 129, 130, 131, 132, 152].

2.1 Search space

The set of solutions of an optimization problem (1), which may be potentially visited in a metaheuristic algorithm, is called the *search space*. If generating feasible solutions is easy, then it is reasonable to define the feasible region F as the search space. On the other hand, if generating feasible solutions is not easy (i.e., searching solutions in F is difficult), we may define an appropriate search space $\tilde{F}$ by taking the problem structures into account. If we adopt a search space different from F, we need to modify the objective function of (1) so that we can evaluate the amount of infeasibility of given solutions as well. Let us denote such a function by $\tilde{f}$. It is preferable to define $\tilde{F}$ and $\tilde{f}$ in such a way that an optimal solution in $\tilde{F}$ with respect to $\tilde{f}$ is also an optimal feasible solution of the original problem (1).

For example, generating feasible solutions are easy for SMP, TSP and MAXSAT, since any permutation of n jobs (resp., cities) is a feasible solution for SMP (resp., TSP), and any n-dimensional 0-1 vector is a feasible assignment for MAXSAT. On the other hand, the problem of finding a feasible coloring with a given number of colors for GCP is known to be NP-complete. In this case, we have to consider how to define $\tilde{F}$ and $\tilde{f}$.

For GCP, the following definitions are possible among others.

(i) Fix the number of colors to k. Then $\tilde{F}$ is the set of all solutions (i.e., colorings) with k colors. Define $\tilde{f}$ to be the number of edges having the same colors on their end vertices. Then a solution x is a feasible k-coloring if and only if $\tilde{f}(x) = 0$. In this case, various numbers of colors k should be tried.

(ii) Consider the number of colors k as a variable. Given any coloring x, let $\tilde{f}(x)$ be the weighted sum of the number of edges $s(x)$ having the same colors on their end vertices and the number of the used colors $k(x)$. The search space $\tilde{F}$ is the set of all colorings to vertices.

(iii) Assume that the colors are indexed with 1, 2,..., and let the search space $\tilde{F}$ be the set of all permutations of n vertices. A feasible coloring is generated from each permutation by assigning colors to the vertices in the order of the permutation, where each vertex receives the minimum color among those feasibly assignable to it. Then $\tilde{f}$ is the number of colors used in the obtained feasible coloring.

In (ii), the weight of $s(x)$ should be sufficiently large to maintain feasibility. This can also be attained by using the lexicographical order (smaller the better) of vector $(s(x), k(x))$ as $\tilde{f}$, which is equivalent to the weighted sum of (ii) with the weight of $s(x)$ infinitely large. Note that the lexicographical order of vector $(k(x), s(x))$ is

not appropriate, since this is equivalent to the weighted sum with the weight of $s(x)$ arbitrarily close to 0. Usually, if the search space include infeasible solutions, the function $\tilde{f}$ and the search strategy should be more carefully designed to make the search smooth and effective. We will give more details on these points in Sections 3.2 and 3.3.

As seen in these examples, there are usually many possible definitions of the search space and the evaluation function. To design efficient metaheuristic algorithms, therefore, it is crucial to use appropriate $\tilde{F}$ and $\tilde{f}$. However, the appropriateness depends on the structure of the given problem, and the general discussion appears to be difficult. In the rest of this paper, it is assumed that we are already given the search space $\tilde{F}$ and the evaluation function $\tilde{f}$ unless otherwise stated. Note that these should be provided by the designers of the metaheuristic algorithms. We also assume that the lexicographic order is used if $\tilde{f}$ is a vector function. Note that the objective value of the best feasible solution (i.e., *incumbent value*) must be retained during the search, if $\tilde{f}$ is different from original f.

In GCP, if method (i) or (ii) is adopted, the operation of changing the color of one vertex is often used. In this case, the change in $\tilde{f}$ can be evaluated efficiently by scanning only the edges incident to the vertex whose color is modified. On the contrary, if method (iii) is adopted, the evaluation of the change in $\tilde{f}$ may take much time, since the change in the coloring generated from the permutation might be large even if the change in the permutation is small. In this sense, method (iii) is considered to be inferior to the other two from the view point of efficiency; however, for scheduling problems with many complicated constraints, this idea is known to be quite effective and is called the *list scheduling* (e.g., [117]).

2.2 Basic strategies

Among basic strategies of approximate algorithms for combinatorial optimization problems are

- greedy method and
- local search (LS).

The greedy method is a one-path algorithm that constructs a feasible solution step by step, on the basis of the local effectiveness. LS starts from an initial solution x and repeat replacing it with a better solution x' (i.e., $\tilde{f}(x') < \tilde{f}(x)$) in its *neighborhood* $N(x)$ until no better solution is found in $N(x)$, where $N(x)$ is a set of solutions obtainable from x by slight perturbations. The resulting solution x, which cannot be improved by a solution in $N(x)$, is called *locally optimal*. The use of the greedy method and LS has a long history, and one of the early applications of them is found in the paper by Croes [32], where the search starts from an initial solution generated by a greedy method, improves it by using LS, and then modifies it by using a complicated algorithm

to obtain an exact solution. It might be interesting to mention that Croes solved a 42-city TSP instance by hand using about 70 hours.

2.3 Multi-start local search

In the multi-start local search (MLS), LS is repeated from a number of initial solutions and the best solution found during the entire search is output. This is one of the most commonly used techniques for combinatorial optimization problems [90, 101, 102, 136, 135]. The initial solution may be generated randomly or by using greedy methods. Some important ideas of using previous search history (1) to generate new initial solutions [135], or (2) to speed-up the search [101, 102] appeared in early literature.

2.4 Genetic algorithm

The genetic algorithm (GA) is a probabilistic algorithm, whose idea comes from evolution. GA repeatedly applies the operations such as *crossover*, *mutation* and *selection* to the set of candidate solutions P. GA starts from an initial candidate solutions P and repeat the following two steps.

1. Generate new solutions $Q \subset N(P)$, where $N(P)$ is defined to be the set of solutions obtainable from P by crossover and mutation operators.
2. Choose $P' \subset P \cup Q$ according to the selection rule, and set $P := P'$.

A crossover operator generates one or more solutions by combining two or more candidate solutions, and a mutation operator generates a solution by slightly perturbing a candidate solution.

GA was proposed by Holland [76]. See, e.g., [11, 34, 68, 94, 115, 133, 149] for more details.

2.5 Simulated annealing

Simulated annealing (SA) is a variant of LS, in which test solutions are randomly chosen from $N(x)$ and accepted with probability that is 1 if the test solution is better than x, and positive even if the test solution is worse than x. The concrete rules are given as follows.

1. Generate a solution $x' \in N(x)$ randomly.
2. Set $\Delta := \tilde{f}(x') - \tilde{f}(x)$ ($\Delta > 0$ implies that x' is a worse solution).
3. Set $x := x'$ with probability 1 if $\Delta \leq 0$, and $\mathrm{e}^{-\Delta/t}$ otherwise, where t is a parameter.

By giving a positive probability to a move to a worse solution, the search is able to escape from poor locally optimal solutions. The acceptance probability is judiciously controlled by a parameter t called *temperature*, whose idea stems from the physical

annealing process. Temperature t is usually set to large (i.e., the probability of moving to a worse solution is high) in the beginning of the search, and gradually decreased as the search proceeds. The behavior of SA becomes the same as that of ordinary LS as $t \to 0$. SA was proposed by Kirkpatrick et al. [92] and Černý [24]. Extensive computational results are found in the series of papers by Johnson et al. [86]. See [1, 3] for more details of SA.

2.6 Tabu search

Tabu search (TS) tries to enhance LS by using the memory of the previous search. Basically the best solution in $N(x) \setminus (\{x\} \cup T)$ is chosen as the next solution, where the set T, called *tabu list* (or *short term memory*), is a set of solutions which includes those solutions most recently visited. Within this restricted neighborhood, a move to a new solution is always executed even if the current solution is already locally optimal. In this process, cycling of a short period can be avoided as a result of introducing T. Another type of memory, called *long term memory*, is often employed in the framework of TS, which memorizes the past search information such as the frequency that each decision variable has been changed, the length of the term that each decision variable takes a certain value and so on. This memory is used to direct the search to the unvisited region (i.e., *diversification*).

TS was proposed by Glover [54, 55]. See [59, 62, 63, 64, 75] for details of TS.

3 General framework of metaheuristics

In this section, we explain essential ideas used in metaheuristic algorithms within a generalized framework of local search. In this paper, we regard a metaheuristic algorithm as a method of repeating the following two basic steps.

I. Generate initial solutions x.

II. Improve x by using (generalized) LS.

A basic framework of LS of Step II was already given in Section 2.2. Here we define LS in the following more general framework.

LS: Start from the given initial solution x, and repeat replacing x with the solution in its neighborhood $N(x)$ (i.e., a move) determined by a specified move strategy, until some stopping criterion is satisfied.

We call the basic LS explained in Section 2.2 *simple LS* if we want to distinguish it with more general LS. To implement algorithm LS, therefore, we must specify:

A. the neighborhood $N(x)$,

B. the evaluation function $\tilde{f}$ of solutions,

C. the move strategy and

D. the stopping criterion.

Rule A specifies how to generate candidate solutions from the current solution. Rule B specifies the goodness criterion for the solutions in the neighborhood. Rule C specifies the order of solutions to be searched in the neighborhood, and the rule to choose the next solution. Rule D specifies when algorithm LS terminates. For example, in the simple LS, the rule for C is to set $x := x'$ if $\tilde{f}(x') < \tilde{f}(x)$ holds, and the rule for D is to stop if x is locally optimal (i.e., $\tilde{f}(x') \geq \tilde{f}(x)$ for all $x' \in N(x)$).

Then most ideas of metaheuristics are considered as the contrivance on the above constituents I, II-A, II-B, II-C and II-D. According to this observation, we will explain various metaheuristic ideas in the following subsections. Note that ideas in II-A (the neighborhood) will be explained separately in Section 5, as they are usually rather complicated.

3.1 I: Initial solutions

One of the simplest ways of generating initial solutions is to use random solutions. This results in the random multi-start local search (MLS) as already explained in Section 2.3. However, it is expected that the solution quality is improved if the search is started from better solutions. The following ideas are mainly based on such consideration.

The greedy randomized adaptive search procedure (GRASP) uses the initial solutions which are generated by randomized greedy method. In the greedy method, a feasible solution is usually constructed step by step by choosing the element with the best local evaluation. Although better initial solutions than random ones are usually obtained, the variety of solutions constructed by this method is quite limited, which is not desirable in the framework of MLS. To overcome this, in GRASP, a feasible solution is constructed, in each step, by randomly choosing an element from the candidate list composed of those elements having good evaluations.

GRASP was proposed by Feo et al. (e.g., [44, 47]), and applied to various combinatorial optimization problems by themselves and others, e.g., [43, 46, 97, 99, 137]. See [45] for details. The basic idea of randomized greedy method has appeared in early papers such as [74], though LS was not incorporated.

In the iterated local search (ILS), the initial solutions are generated by slightly perturbing a good solution $x_{\rm seed}$ found during the past search. The initial solutions are usually generated by randomly choosing a solution in the neighborhood $N'(x_{\rm seed})$, where a different neighborhood from N of LS is often used. ILS was proposed by Johnson [85]. While the incumbent solution (i.e., the best feasible solution found so far) is used as $x_{\rm seed}$ in the basic ILS of [85], there are variants of ILS, called *chained local optimization* or *large-step Markov chain*, in which $x_{\rm seed}$ is chosen randomly according to the rule similar to SA [104, 108, 109, 110]. (Historically the preprints of Martin et al. [109] inspired Johnson to propose ILS, a simplified variant of Martin–Otto–Felten

algorithm, as described in p. 292 of [87].)

A variant of ILS, in which the size of the neighborhood N' for random perturbations is adaptively changed, is called the variable neighborhood search (VNS) [19, 20, 73, 112]. Usually, in VNS, the size of N' is initially set to be small, gradually made larger if no improved solution is found, and reset to its initial size if improvement occurs. (Note that the neighborhood N for LS is kept to be the same in general.)

A variant of GA, in which solutions generated by the crossover and mutation operators are used as the initial solutions for LS, is called the genetic local search (GLS). While only one solution is used for the generation of initial solutions in ILS, GLS maintains many candidate solutions and generate initial solutions by combining them. (As the word genetic algorithm (GA) is also used to mean the general framework including GLS, we use *simple GA* to denote the genetic algorithms which do not incorporate LS, if we want to distinguish them from GLS.) The basic idea of GLS was proposed in [18]. Early references such as [34, 68, 82, 83, 111, 113, 114, 146, 151] have also mentioned the idea of GLS. To our knowledge, the word genetic local search first appeared in [151]. A similar but more general mechanism of generating initial solutions from many reference solutions is also proposed and is called the *scatter search* [53, 56, 57].

While good solutions found in the past are used to generate initial solutions in GLS, there are variants of GLS in which partial structures or statistical information of good solutions are used [17, 147]. One of such approaches, called the *ant system* (or *ant colony system*), is inspired by the behavior of ants, which are able to find good solutions to shortest path problems between a food source and their home colony by communicating via *pheromones*. In the ant system, initial solutions are generated by using greedy methods, in which local evaluation is adaptively modified by using the information of pheromones. During the search, pheromones are laid on the elements (e.g., tour edges for TSP) which are included in good solutions so that good elements are more likely to be chosen in the initial solutions. This method was proposed in [30, 36, 38]. Though LS was not incorporated in early references of ant system, better results are reported by incorporating LS into the framework of ant system [37, 50, 105, 145].

3.2 II-B: Evaluation functions of solutions

A simple way of evaluating solutions is to use the original objective function f as $\tilde{f}$. However, this method can not be used if the search space $\tilde{F}$ includes infeasible solutions. In such cases, penalty terms are often added to evaluate the degree of infeasibility. It is also effective to dynamically change the evaluation function $\tilde{f}$ during the search. Among such approaches are the *noising* method, *perturbation*, *search space smoothing method*, *guided local search*, long term memory of TS and so on.

The noising method [26] (or perturbation [29]) uses a randomly perturbed $\tilde{f}$ and

the original $\tilde{f}$ alternately. In the noising method, the amount of randomness added to $\tilde{f}$ is initially set to be large, and then gradually decreased until it becomes 0 at the last iteration. The initial solution for LS with a perturbed $\tilde{f}$ is either the incumbent solution or the previous locally optimal solution with respect to the original $\tilde{f}$. The initial solution for LS with the original $\tilde{f}$ is the locally optimal solution of the previous LS, in which a perturbed $\tilde{f}$ is used. The method of perturbation is slightly different from the noising method in that 1) the amount of perturbation is fixed or depends on the quality of the current solution, and 2) the initial solution for LS with perturbed $\tilde{f}$ is always the incumbent solution. If the steps of LS with perturbed $\tilde{f}$ is regarded as a process to generate initial solutions of the subsequent LS with the original $\tilde{f}$, then these methods can be considered as variants of ILS.

In the search space smoothing method [69, 70], the evaluation function $\tilde{f}$ is changed deterministically. For a parameter α (≥ 1), function $\tilde{f}^{(\alpha)}$ is defined so that $\tilde{f}^{(1)} = \tilde{f}$ and $\tilde{f}^{(\infty)}(x) = c$ ($\forall x \in \tilde{F}$) hold, where c is any constant. For TSP, for example, $\tilde{f}^{(\alpha)}$ is defined as follows [70]. Here we assume that the distance d_{ij} satisfies $0 \leq d_{ij} \leq 1$ for all i and j. Let $\bar{d} = \sum_{i \neq j} d_{ij}/(n(n-1))$ and

$$d_{ij}^{(\alpha)} = \begin{cases} \bar{d} + |d_{ij} - \bar{d}|^{\alpha}, & d_{ij} \geq \bar{d} \\ \bar{d} - |d_{ij} - \bar{d}|^{\alpha}, & d_{ij} < \bar{d}. \end{cases}$$

Then $\tilde{f}^{(\alpha)}$ is defined to be the sum of $d_{ij}^{(\alpha)}$ of tour edges $\{i, j\}$. The value of α is initially set to be large, and then gradually decreased until it becomes 1 at the last iteration. By this rule, the solution structures which affect the cost in large amount are captured in the initial stage, and then detailed structures are gradually considered in the later steps.

In the guided local search, the element (e.g., tour edges of TSP) with the largest cost included in the locally optimal solution x in the last trial is penalized. In the next search, the evaluation function $\tilde{f}$ is modified based on such penalty values of elements and x is used as the initial solution of LS. If $\tilde{f}$ is appropriately modified, x is no longer locally optimal under the new evaluation function, and the next LS resumes from x. By this rule, we can search those solutions around x, which do not contain the element that incur the largest cost. This method was proposed in [153], and applied to various combinatorial optimization problems [91, 150, 154]. A similar and slightly simpler idea, called the *weighting strategy*, is also proposed for the satisfiability problem (SAT) [142].

Similar ideas are also used in TS as a part of the long term memory. Various other useful ideas to make use of the search history, such as the frequency of changing a certain variable, are also proposed within the framework of the long term memory [63]. One of such ideas is to avoid long term cycling by penalizing moves which have changed their values frequently. The long term memory is also used to guide the search into unvisited region (i.e., diversification), as well as to intensify the search around good

solutions found during the previous search. The objective of the latter is the same as those ideas in Section 3.1. Thus the long term memory includes broad range of ideas of both intensification and diversification, where the meaning of intensification and diversification will be explained in Section 5.1.

If infeasible solutions are included in the search space, the weighted sum of the objective function f and the penalty to infeasibility (or the vector consisting of the penalty term and f) is usually used as the evaluation function $\tilde{f}$ as explained in Section 2.1. In such cases, the performance of the search is often improved by using small penalty weights that encourage the visit to infeasible region more frequently. For problems with many constraints, it is also effective to give penalty weights to individual constraints and tune them separately. The use of the penalty function method is not limited to combinatorial optimization. It has been used in wide rage of optimization problems for long time. Here we mention some recent applications of this method to combinatorial optimization [52, 86, 164, 165, 163]. Note that feasible solutions will not be found if the penalty weights are too small. To avoid this, the penalty weights are adaptively controlled in some applications (e.g., [52, 163]) by gradually increasing the penalty weights on those constraints not satisfied for long time.

Evaluation functions $\tilde{f}$ need not be directly related to the objective function f. For example, Johnson et al. (part II of [86]) used the following function for GCP, where the number of colors k is considered to be a variable and the search space is the set of all color assignments to vertices. For a given solution x, let V_i be the set of vertices colored with i and E_i be the set of edges both of whose end vertices are in V_i, $i = 1, \ldots, k$. Then the evaluation function is defined as

$$\tilde{f}(x) = -\sum_{i=1}^{k} |V_i|^2 + \sum_{i=1}^{k} 2|V_i| \cdot |E_i|, \tag{2}$$

which is to be minimized. By the first term of (2), a large color set V_i tends to get more vertices than smaller ones, and as a side effect of this, the number of colors k is minimized. The second term of (2) is the penalty on those edges having the same colors on the end vertices. Compared to the straightforward evaluation function such as method (ii) of Section 2.1 (i.e., the weighted sum of k and $\sum_{i=1}^{k} |E_i|$), a merit of using (2) is that we can evaluate the gain of a move in which the number of colors is not changed but the size of a small V_i is decreased. If the neighborhood of x is defined to be the set of solutions obtainable by changing the color assignment of one vertex, then locally optimal solutions are always feasible (i.e., $E_i = \emptyset$ for all i). To see this, suppose that E_i is nonempty, and let v be an end vertex of an edge in E_i. If the color of v is changed to $k + 1$ (i.e., a new color), then the second term of (2) is reduced by at least $2|V_i|$ while the first term is increased by at most

$$|V_i|^2 - ((|V_i| - 1)^2 + 1^2) = 2|V_i| - 2.$$

Note that this property also means that the value of (2) can always be reduced at the cost of increasing k if the current solution x is infeasible. Therefore the move strategy should be designed so that those moves that reduce (2) without increasing k are given priority, if such moves exist.

3.3 II-C: Move strategies

One of the basic move strategies is to choose an improved solution in the neighborhood. (Note that the evaluation function $\tilde{f}$ is not necessarily equal to f and has a wide variety as explained in Section 3.2.) In this case, two strategies, called the *first admissible move* strategy and the *best admissible move* strategy, are often used. In the first admissible move strategy, solutions in $N(x)$ are scanned according to a prespecified order, and the first improved solution found is immediately accepted as the next solution. In this strategy, the order of searching the neighborhood may affect the computational time and solution quality. In the best admissible move strategy, the best solution in $N(x)$ is chosen as the next solution.

Move strategies which allow moves to nonimproved solutions are used in the metaheuristic framework such as simulated annealing (SA) and tabu search (TS) of Sections 2.5 and 2.6, respectively. Here we explain two variants of SA called the threshold accepting (TA) and the great deluge algorithm (GDA), some additional rules of tabu search (TS), and a simple move strategy of an algorithm called WALKSAT.

In TA, Step 3 of SA is replaced by

3′. Set $x := x'$ if $\Delta < \tau$ holds,

where τ is a parameter called the *threshold*. Threshold τ is usually set large in the beginning of the search, and is gradually decreased as the search proceeds, as in the case of temperature t of SA. While a random number must be generated and the exponential function is calculated in Step 3 of SA, TA does not need to compute them and is slightly more efficient and simpler. TA was proposed by Dueck and Scheuer [41]. Some comparative results with other metaheuristic algorithms are found, e.g., in [4, 100, 151].

GDA is also a variant of SA. For convenience, we assume the maximization of $\tilde{f}$ in explaining GDA. In GDA, Steps 2 and 3 are replaced by

2′. Set $x := x'$ if $\tilde{f}(x') \geq \omega$ holds,

where ω is a parameter called the *water level*. Initially, the value of ω is set to be sufficiently small and is gradually increased to an appropriate value between $\tilde{f}(x')$ and the current ω whenever $\tilde{f}(x') > \omega$ holds. The algorithm is motivated by the situation where a person wants to escape from increasing water of a great deluge to a mountain as high as possible. GDA was proposed in [40], and comparisons with other metaheuristics are found, e.g., in [144].

While TS usually chooses the best solution in $N(x) \setminus (\{x\} \cup T)$ (T is the tabu list)

as explained in Section 2.6, there is a variant of this called the *candidate list strategy*, which chooses the best solution among the candidate solutions that are considered to be promising. It is also possible to use the first admissible move strategy if an improved solution is found in $N(x) \setminus (\{x\} \cup T)$, and use the above rules otherwise. Moreover, it is sometimes allowed to choose a solution $x' \in N(x)$ as the next solution, even if x' is in the tabu list T (i.e., $x' \in N(x) \cap T$), provided that x' is considered to be worth choosing, e.g., $\tilde{f}(x')$ is the best among those solutions previously visited by the search. This is called the *aspiration criteria*.

There is a method called *strategic oscillation*, which is usually used within the framework of TS. This method is adopted only if the search space includes infeasible solutions. In this method, the search is judiciously controlled to visit the feasible and infeasible regions alternately by adaptively using different types of neighborhoods. Strategic oscillation is known to be effective to those problems where simple LS will not work well, e.g., the graph coloring and knapsack problems [60, 61, 71]. Many ideas of this kind are proposed under the framework of TS. See [63] for more details.

The move strategy of algorithm *WALKSAT* for the satisfiability problem (SAT) [143] uses an idea similar to the candidate list strategy. This algorithm is specialized to problem SAT, but its idea is applicable to other problems. In this method, the search space is the set of all 0-1 assignments, the neighborhood is the set of solutions obtainable from the current solution by flipping the assignment of one variable (i.e., neighborhood N_1 whose definition will be given in Section 4.5), and the evaluation function $\tilde{f}$ is the number of unsatisfied clauses. Then the next solution is chosen by the following simple rule. First a clause i with $C_i(v) = 0$ (i.e., an unsatisfied clause) is randomly chosen, and then the variable to be flipped is chosen from those whose corresponding literals are in clause i by (A) choosing the best flip or by (B) choosing a random flip, where rules (A) or (B) is chosen randomly for each move. In either rule, the probability of finding an improved solution (if it exists) is positive, since at least one unsatisfied clause is satisfied by the move. Moreover, the number of candidates tested for each move is at most the number of literals in clause i, which is usually much smaller than n. As a result, many moves can be tried within a short time. Though the rules are simple, the algorithm is known to be quite effective [143]. Now suppose that the instance is satisfiable and let v^* be a satisfying assignment (i.e., $\sum_{i=1}^{m} C_i(v^*) = m$). One of the important features of this method is that, if rule (B) is used, the probability of getting closer to v^* is at least $1/l_i$, where l_i denote the number of literals in clause i, since at least one variable in clause i must have different assignment in v^*. By this property, it is known that, if $l_i \leq 2$ hold for all i and the instance is satisfiable, then algorithm WALKSAT with rule (B) will find a satisfying assignment in $O(n^2)$ time with probability arbitrary close to one [124]. This property motivated the proposal of algorithm WALKSAT for general SAT instances.

3.4 II-D: Stopping criteria

The stopping criterion of each LS should be determined by considering the rules of II-C (the move strategy). If the search always moves to a better solution, then the algorithm is usually terminated at a locally optimal solution. On the other hand, if the search can be continued as long as we want, as in the cases of SA and TS, the following rules are often used: (1) Stop if the number of moves and/or computational time reaches a prespecified limit, (2) Stop if the incumbent value (or the best value of $\tilde{f}$) has not been improved for a prespecified number of moves. Other rules involving algorithmic parameters are also possible. For SA, for example, as temperature t is gradually decreased from the initial value, we can stop if t becomes smaller than a prespecified value.

4 Metaheuristics as simple optimization tools

In this section, we compare representative metaheuristic algorithms for problems SMP and MAXSAT. As discussed in Section 1, the objective of this experiment is to observe general tendencies of metaheuristics as simple optimization tools.

All tested algorithms were coded in C language and run on workstations Sun SPARC station IPX for SMP and Sun Ultra 2 Model 2300 for MAXSAT. The quality of the obtained solutions is evaluated by the average error from the best cost values, which were found in the entire experiment. (In the case of MAXSAT, the average error from the optimal values are used.) The efficiency of algorithms is measured by the number of the solution samples evaluated, rather than the computational time, since the computational time depends on the computers used and other factors such as programming skill.

4.1 Single machine scheduling problem

For the cost $g_i(c_i)$ on the completion time c_i of job i, we use

$$g_i(c_i) = h_i \max\{d_i - c_i, 0\} + w_i \max\{c_i - d_i, 0\},$$

where $d_i \in Z_+$ is the due date of job i, and $h_i, w_i \in Z_+$ are respectively the weights given to earliness and tardiness of job i. The SMP is known to be NP-hard for most of the interesting forms of $g_i(c_i)$ including the above case. Ten problem instances for $n = 100$ are generated as described in [81].

4.2 Metaheuristic algorithms for SMP

We implemented algorithms MLS, GRASP, ILS, GA, GLS, SA, TA, GDA and TS for SMP, which are explained in this section. More details of them are found in [159].

For algorithm MLS, the initial solutions of Step I (of the metaheuristic framework in Section 3) are generated randomly and the simple LS is used for Step II. (Other metaheuristic algorithms also use these rules unless otherwise stated.) For the neighborhood of II-A, we examined $N_{\rm ins}(\sigma)$ = {the set of solutions obtainable from σ by changing the position of one job} (the *insertion* neighborhood) and $N_{\rm swap}(\sigma)$ = {the set of solutions obtainable from σ by exchanging the positions of two jobs} (the *swap* neighborhood). The objective function $cost(\sigma)$ is used as the evaluation function $\tilde{f}$ of II-B. For II-C, we tested two move strategies, the first admissible (abbreviated as FA) and the best admissible (abbreviated as BA). It was observed in the preliminary experiment that FA obtains better solutions much earlier than BA. It was also observed in [156] that the differences in the average quality of locally optimal solutions is small between the two strategies. These results indicate that FA obtains much more number of locally optimal solutions, where solution quality of each locally optimal solution is comparable with those obtained by using BA, within the same number of solution samples. This is the reason why FA shows better performance. Based on this, the search strategy is fixed to FA in the following experiments.

For algorithm GRASP, various ideas can be incorporated in the greedy algorithm to generate initial solutions. We tested 12 greedy methods which are used in [156].

For algorithm ILS, various rules are possible in (1) the selection of the solution $\sigma_{\rm seed}$ used to generate the next initial solution for LS, and (2) the perturbation added to solution $\sigma_{\rm seed}$ (i.e., mutation). Usually the solution $\sigma_{\rm seed}$ is chosen from between the previous $\sigma_{\rm seed}$ and the locally optimal solution of the previous LS, which is denoted $\sigma_{\rm last}$. We adopt the rule used in [110], which is explained as follows.

(1) Set $\Delta := cost(\sigma_{\rm last}) - cost(\sigma_{\rm seed})$, and then set $\sigma_{\rm seed} := \sigma_{\rm last}$ if $\Delta \leq 0$; otherwise set $\sigma_{\rm seed} := \sigma_{\rm last}$ with probability $\mathrm{e}^{-\Delta/t}$, where parameter t is fixed during the search.

(2) The mutation employed in our experiment perturbs a candidate solution σ by randomly choosing $\sigma' \in N(\sigma)$ and setting $\sigma := \sigma'$. Two types of neighborhood $N_{\rm ins}(\sigma)$ and $N_{\rm swap}(\sigma)$ are used as $N(\sigma)$; the resulting mutations are denoted as INS and SWAP respectively.

For algorithms GA and GLS, various rules of crossover, mutation and selection are possible. We examined the crossover operators such as order crossover [33, 114], partially mapped crossover [67], cycle crossover [120] and so on. The comparisons of these crossover operators are found in [156]. For the mutation operator, we examined both operations INS and SWAP as in the case of ILS. In our experiment, the set of generated candidate solutions $Q \subseteq N(P)$ is determined as follows. We use $|Q| = 1$ and the child $\sigma_C \in Q$ is obtained by randomly selecting two parents σ_A, $\sigma_B \in P$, mutating either σ_A or σ_B, and then applying crossover of σ_A and σ_B. Then taking the solution $\sigma_{\rm worst}$ that satisfies $cost(\sigma_{\rm worst}) \geq cost(\sigma)$ for all $\sigma \in P \cup Q$, we let $P' :=$

$P \cup \{\sigma_C\} \setminus \{\sigma_{\rm worst}\}$. Note that the selection is executed only if the child $\sigma_C \in Q$ is not in P.

For algorithm SA, various rules are possible to control temperature t. The rules used in our experiment is similar to the one in [86]. Our algorithm includes four parameters $p_{\rm init}$, $\alpha_{\rm freeze}$, $\alpha_{\rm loop}$ and $\beta_{\rm temp}$. The initial temperature t is determined for an initial solution σ so that the probability of moving from σ to a nonimproved solution is approximately $p_{\rm init}$. Then Steps 1 to 3 of Section 2.5 is repeated $\alpha_{\rm loop} \cdot |N|$ times with a fixed temperature t, and then t is decreased by $t := \beta_{\rm temp} \cdot t$. This loop is repeated until no improved solutions are found during $\alpha_{\rm freeze} \cdot |N|$ consecutive solution samples. Upon termination, the search is restarted from a randomly chosen initial solution unless a sufficient number of solution samples has been tested.

For algorithm TA, we used the same rules as SA to control threshold τ.

For algorithm GDA, various rules to control water level ω is possible. Here we use parameter $\beta_{\rm rain}$ ($0 < \beta_{\rm rain} \leq 1$) called the *rain speed*. For an initial solution σ, the initial water level is set to $\omega := cost(\sigma)$. Then, for each iteration, we randomly choose a solution σ' from $N(\sigma)$ and then set $\omega := \omega - \beta_{\rm rain}(\omega - cost(\sigma'))$ if $cost(\sigma') < \omega$ holds. The stopping criterion is the same as SA.

For algorithm TS, move strategies and definitions of tabu list T have to be determined. We scan $N(\sigma)$ and select the first solution σ' satisfying $cost(\sigma') < cost(\sigma)$ and $\sigma' \notin \{\sigma\} \cup T$, or satisfying $cost(\sigma') < cost(\sigma_{\rm best})$ (i.e., aspiration criterion), where $\sigma_{\rm best}$ is the cost of the best solution found during the past search. If none of the solutions in $N(\sigma)$ is selected by the above rule (i.e., σ is locally optimal with respect to the neighborhood restricted by T), the best solution in $N(\sigma) \setminus (\{\sigma\} \cup T)$ is chosen as the next solution. Two types of tabu lists $T_{\rm job}$ and $T_{\rm pos}$ are considered, where $T_{\rm job}(\sigma) = \{\sigma' \in N(\sigma) \mid$ the move from σ to σ' changes the position of a job whose position has been changed in the last $t_{\rm tabu}$ moves$\}$ and $T_{\rm pos}(\sigma) = \{\sigma' \in N(\sigma) \mid$ the move from σ to σ' assigns a job to the position where it has been assigned in the last $t_{\rm tabu}$ moves$\}$. The parameter $t_{\rm tabu}$ is a prespecified nonnegative integer called *tabu tenure*. We also use the long term memory, such as the number of times job i was changed its position from position k and the length of the term (counted by the number of moves) job i was assigned to position k, to penalize those solutions that potentially cause cycling in the long time period.

4.3 Comparison of metaheuristics for SMP

All the metaheuristic algorithms explained in Section 4.2 are compared in this section. Figures 4.3 and 4.3 show how the average error (%) of the best solutions improves as the number of samples increases for problem instances with $n = 100$. Both neighborhoods $N_{\rm ins}$ (Figure 4.3) and $N_{\rm swap}$ (Figure 4.3) were examined, where in the case of GA, mutation operators INS and SWAP were used corresponding to $N_{\rm ins}$ and $N_{\rm swap}$,

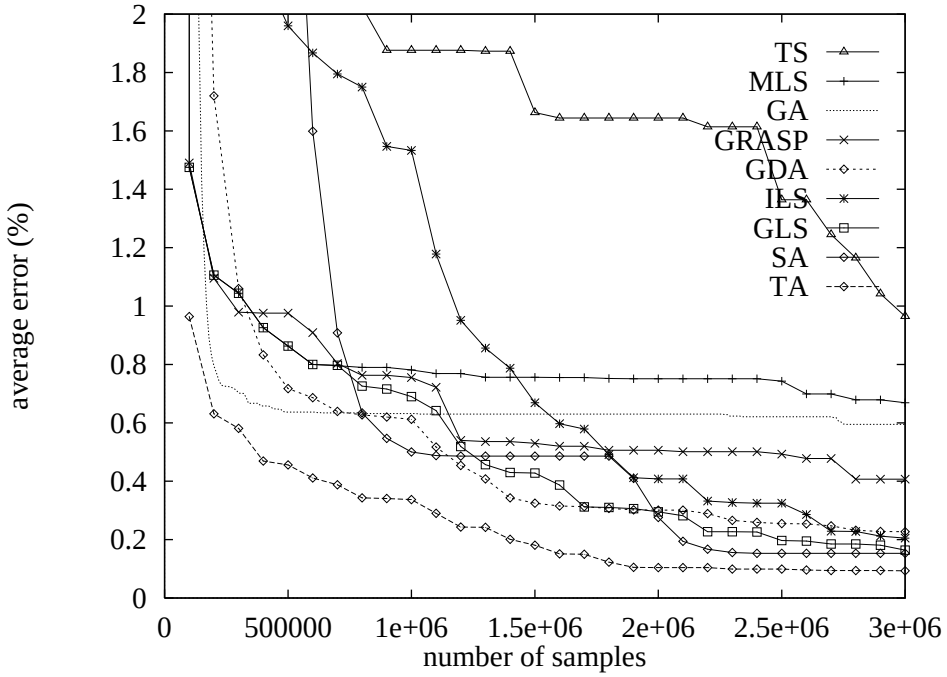

Figure 1. Average error in % against the number of samples (neighborhood N_{ins} and mutation INS are used)

respectively. Parameter values of the tested algorithms were tuned by preliminary experiments and the best values among those tested were used.

From these results, we can observe the following.

(1) The quality of solutions obtained by GA is rather poor.

(2) GRASP, ILS, GLS, SA, TA and GDA improve the performance of MLS further, among which ILS, GLS, SA, TA and GDA appear more powerful if the same neighborhood is used.

(3) Performance of the tested algorithms except GA critically depends on the type of neighborhood, and the results with N_{swap} is better than N_{ins}.

4.4 Maximum satisfiability

The instances of MAXSAT used in our experiments are the random instances used in [138] with $n = 100$ and $m = 850$, which were taken from the web site of Resende[2]. They were generated from DIMACS[3] benchmark instances called `jnh`, whose generation rules are found in [77], by giving random weights from integers in [1, 1000] to clauses.

[2]http://www.research.att.com/~mgcr/data/maxsat.tar.gz

[3]ftp://dimacs.rutgers.edu/pub/challenge/sat/benchmarks/cnf/

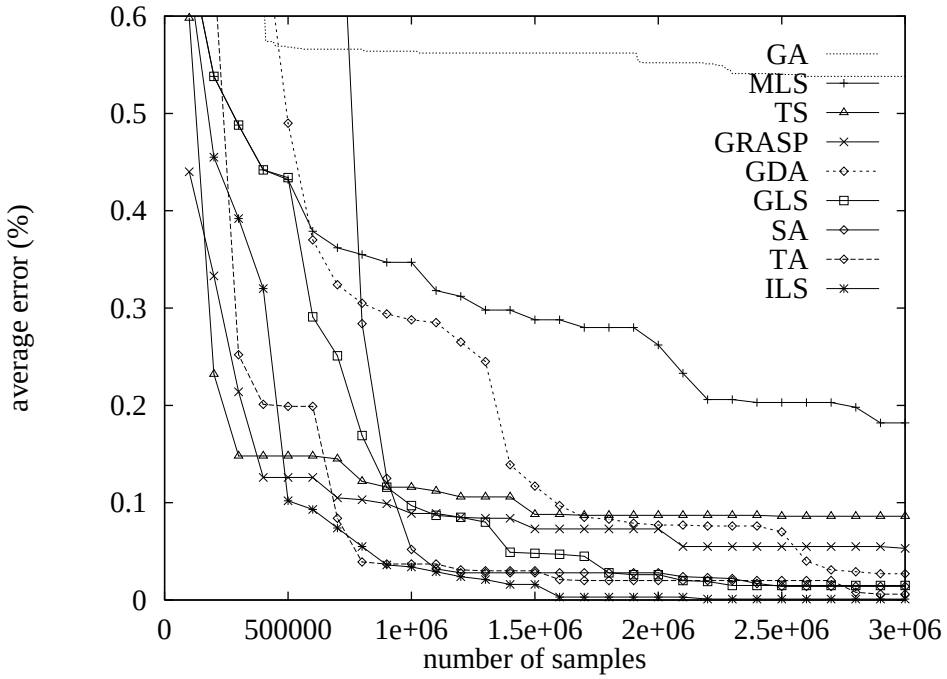

Figure 2. Average error in % against the number of samples (neighborhood N_{swap} and mutation SWAP are used)

4.5 Metaheuristic algorithms for MAXSAT

We implemented algorithms MLS, GRASP, ILS, GA, GLS, SA, TA and TS for MAXSAT which are explained in this section. As their frameworks are basically the same as those for SMP, here we only explain problem dependent parts which are different from SMP.

For algorithm MLS, the *k-flip neighborhood* $N_k(v) = \{v' \in \{0,1\}^n \mid \text{Hamming distance between } v' \text{ and } v \text{ is at most } k\}$ with $k = 1$ and 2 were tested. Among N_1 and N_2, the quality of locally optimal solutions with N_2 was better than that of N_1; however, the LS with N_2 requires long time and the number of restarts (i.e., the number of initial solutions for LS) became quite small within the given number of solution samples if we used N_2. Based on this observation, we decided to restrict our attention to N_1 for all the other metaheuristic algorithms, since, if we use N_2, not much difference in solution quality would be observed among metaheuristic algorithms with such small number of LS trials. As for the move strategy, we tested both FA and BA with MLS, and decided to use only FA for other metaheuristic algorithms, as strategy FA found better solutions faster than BA.

For algorithm GRASP, we tested 4 types of greedy algorithms: those proposed in [84] and [138] and two variants of them. It is known that the quality of a solution obtained by the greedy method of [84] is not worse than the expected value of the objective function $gain(v)$ for a random assignment v in which the 0-1 assignment of each variable is chosen from $\{0,1\}$ with probability 1/2.

For algorithm ILS, the operation of mutation perturbs a solution v by randomly choosing $v' \in N_k$ and setting $v := v'$, where k is a prespecified parameter.

For algorithms GA and GLS, we tested 1-point, 2-point and uniform crossover operators [34, 68]. Note that the set of solutions obtainable by 1- or 2-point crossover operator is different if we use different order of the variable indices, though such order is not directly related to the problem structure. Based on this consideration, we also tested variants of these crossover operators, in which the order of variable indices are randomly generated whenever a crossover is executed. Mutation we used is to perturb a solution v by randomly choosing $v' \in N_1$ and setting $v := v'$.

For algorithm TS, we used the set $T_{\rm var}(v)$ for the tabu list, where $T_{\rm var}(v) = \{v' \in N_1(v) \mid$ the move from v to v' changes the 0-1 assignment of a variable whose assignment was changed during the recent $t_{\rm tabu}$ moves$\}$. We did not use the long term memory for MAXSAT, since TS without it was already quite effective.

4.6 Comparison of metaheuristics for MAXSAT

The metaheuristic algorithms explained in Section 4.5 are compared in this section. Figure 4.6 shows how the average error (%) of the best solutions improves as the number of samples increases for problem instances with $n = 100$ and $m = 850$. Parameter values of the tested algorithms were tuned by preliminary experiments to the best values among those tested.

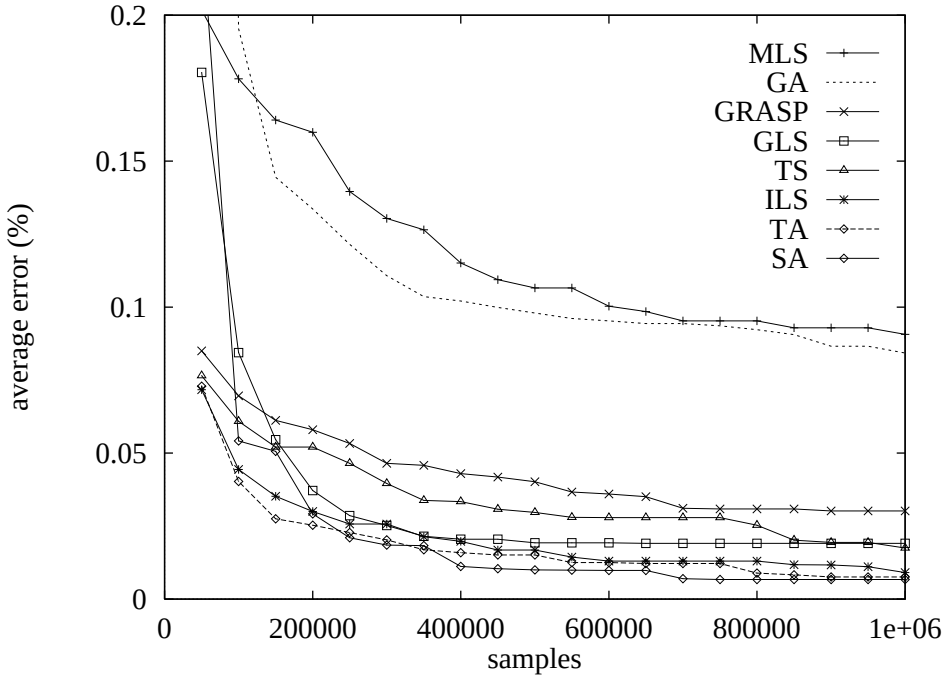

Figure 3. Average error in % against the number of samples ($n = 100, m = 850$)

From Figure 4.6, we can observe the following.

(1) The performance of GA is about the same as MLS.
(2) The performance of all the tested algorithms except GA is better than MLS.

4.7 How to use metaheuristics as simple optimization tools

Summarizing the computational results in Sections 4.3 and 4.6, we observed the following.

(i) The performance of GA is about the same as, or rather worse than MLS.
(ii) The performance of TS crucially depends on the problem structures and the neighborhoods.
(iii) Algorithms ILS, GLS and SA (and its variants) improve the performance of MLS. The differences among them in the performance is small compared to the amounts of improvements from MLS.

The reason for (ii) will be explained later in this section, and the reasons for (i) and (iii) will be explained later in Section 5.1 in connection with the concept of POP (POP is also explained in Section 5.1). It might be difficult to draw general conclusions from the limited computational results on two problems; however, we can expect that similar results will be obtained for a wide range of problems, since the above observations are common to the two tested problems of different nature. Here we note that the behavior of the tested metaheuristic algorithms on various operators and parameter values were also extensively examined, though the detailed results are omitted due to the limited space. The tendencies observed in such experiments will also be mentioned later in this section.

Based on the above observations, as well as other factors such as the time and labor for writing the program codes, we can summarize our recommendation on how metaheuristics are used as *simple* optimization tools.

1. Use MLS first. In this case, try various neighborhoods to improve the performance, before trying other metaheuristic algorithms.
2. If obtaining solutions of higher quality is important, use ILS.
3. If the performance of ILS is not satisfactory, use GLS or SA.

While we mentioned SA in guide 3 as it has an established reputation, its variants such as TA and GDA might be also worth trying, judging from our computational results.

The following are the observations on how the tested metaheuristic algorithms behaved on different operators and parameter values, and how hard the programming was, which may support the above guidelines on the use of metaheuristics. Note that the comments on the programming efforts might be subjective to some extent.

MLS: The implementation of MLS is easy if LS is already available. It was observed that the performance of other metaheuristic algorithms sometimes became worse than MLS if the operators and/or parameter values were not appropriate. Moreover, for SMP, the quality of solutions obtained by MLS with N_{swap} was about the same

as that obtained by algorithms ILS, GLS, SA and so on with $N_{\rm ins}$, whose operators and parameter values were appropriately tuned. In view of these, it is recommended to try MLS first (1) to examine various neighborhoods carefully, and (2) to provide a standard on the performance of other metaheuristic algorithms.

ILS, GLS and SA: These three algorithms all exhibited better performance than MLS, and their performance was relatively robust on various operators and parameter values. The differences among them in the performance is small compared to the amount of improvements from MLS. As for the programming efforts, if the subroutine of LS is already available, ILS and GLS are slightly easier than SA in that the subroutine of LS can be used as a module without any modifications, while we must change the rules on move strategies of LS to implement SA. Among ILS and GLS, ILS is easier, as it handles only one candidate solution, while GLS must handle a set of candidate solutions and more operators (e.g., crossover) must be designed.

GA: The best performance obtained by GA was about the same, or rather worse than MLS, even after tuning various parameters such as $|P|$, and trying various crossover operators. As recognized by many researchers (e.g., [133]), simple GA will not be competitive with other effective metaheuristic algorithms unless LS and/or other heuristic methods are incorporated.

GRASP: The performance of GRASP was sensitive to the choice of the greedy method and the parameter values for generating initial solutions. The quality of solutions obtained by GRASP was better than MLS if the choice of the greedy method and the parameter values were appropriate, but worse otherwise. Moreover, even if the quality of the solutions obtained by a greedy method was better, it did not necessarily lead to better performance of the resulting GRASP algorithm. The best results attained by various different GRASP algorithms were better than MLS, but slightly worse than other metaheuristic algorithms. As for the programming, implementing greedy method was as hard as (or sometimes harder than) implementing LS. In view of these, we did not recommend GRASP in the above guidelines. It should be noted, however, that the average quality of solutions obtained by one trial (i.e., the LS iteration from one initial solution) of GRASP is usually better than that of MLS. This property is preferable if the time limit is short and not many trials of LS can be executed. It is also noted that GRASP algorithms are often quite effective if problem structures are fully exploited in the greedy method and LS, as reported in the literature.

TS: The performance of TS was sufficiently good for MAXSAT, even if a simple implementation without long term memory was used; however, it was sensitive to parameter $t_{\rm tabu}$. For SMP, our TS did not work well, since short term cycling could not be avoided with the simple tabu lists. It might be possible to improve the performance of TS by using more sophisticated tabu lists and/or incorporating various other ideas of TS, but the resulting algorithm will not be simple. Moreover, the basic move strategy

of TS, in which the best solution among the non-tabu neighbors is chosen, usually takes much time if naively implemented. To avoid this, sophisticated data structures are often incorporated in TS to find the next move efficiently. One of the conceivable reasons for the unsatisfactory results of TS is that we did not use such sophisticated data structures, as simplicity was more important in our experiments.

It should be noted that many good results of TS algorithms have been reported, in which various sophisticated ideas are incorporated. This is mainly because the framework of TS is well-suited with advanced techniques, such as those which will be explained in Section 5. In view of these, TS is considered as a tool for advanced users.

There are many other papers in which various metaheuristic algorithms are compared, e.g., [4, 31, 39, 85, 87, 105, 144, 151], whose motivations are different from ours. Note that drawing general conclusions is hard, as the results may be different if the problems used for the experiments are different.

5 Advanced techniques for metaheuristics

If better solutions are necessary than those found by simple metaheuristic algorithms of the previous section, we should pay more attention to "how to exploit the problem structures" than to "which of metaheuristic frameworks are used". Moreover, adaptive tuning mechanisms of important parameters will become necessary if metaheuristic algorithms are used in practical situations, as it is not desirable to tune them for every instance. Before explaining these elements, we will discuss below the basic ingredients of metaheuristics such as the *proximate optimality principle* (abbreviated as POP), and the ideas of *intensification* and *diversification*.

5.1 Basic ingredients of metaheuristics

The concept of POP is the following intuitive understanding on problem structures: "good solutions have many common structures". Most metaheuristic algorithms are based on this idea in that they iteratively generate solutions having similar structures with those good solutions found during the past search. This is an underlying motivation of operators such as the neighborhood operation, mutation and crossover. A key to a successful design of a metaheuristic algorithm is to find such structures that good solutions have in common. For problem TSP, for example, the set of tour edges (the pairs of cities adjacent in the tour) is known to be important. For example, it is reported by Lin and Kernighan in [102] that approximately 85% of tour edges on the average are common to two locally optimal solutions obtained by the LS proposed by them, which is one of the most effective and well-known heuristic algorithms for TSP.

Here we show another example of POP for MAXSAT, in which assignment v_j of

each variable y_j is considered. Figure 5.1 shows the distribution of randomly generated solutions of an instance with $n = 100$ and $m = 850$, where the horizontal axis shows the Hamming distance from an exact optimal solution and the vertical axis shows the error in % from the optimal value. For each value of distance, 1000 solutions were generated randomly. From the figure, we can observe a strong positive correlation between the Hamming distance from the optimal solution and the solution quality. That is, good solutions tend to be located near the optimal solution (i.e., have the common values of assignment v_j); suggesting that POP holds. More experimental results regarding POP for MAXSAT are found in [162]. The correlation between the basic operators and the objective values is also studied by other researchers on different problems [17, 89, 107, 134, 166].

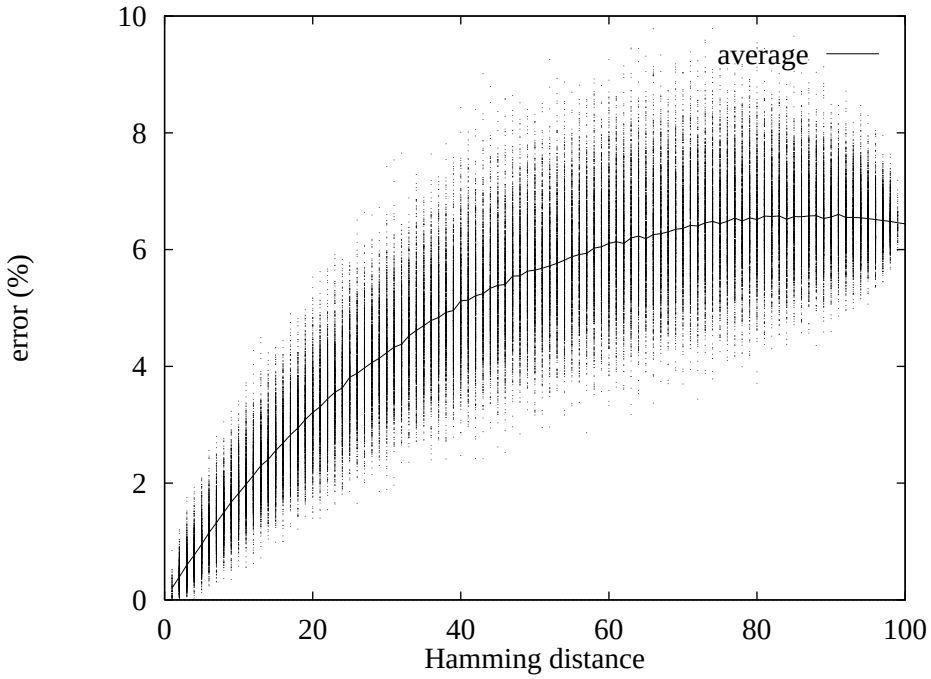

Figure 4. Error in % against Hamming distance from the optimal value

The ideas of intensification and diversification are often mentioned in designing metaheuristic algorithms. Intensification is to search carefully and intensively around good solutions found in the past search (i.e., incarnation of POP). Diversification, on the contrary, is to guide the search to unvisited region. These terminologies are usually used to explain the basic elements of TS, but these are essential to all the metaheuristic algorithms. In other words, various metaheuristic ideas should be understood from the view point of these two concepts, and metaheuristic algorithms should be designed so that intensification and diversification play balanced roles.

The reasons of observations (i) and (iii) in Section 4.7 might be explained as follows based on POP. As for (i), intensification is not sufficiently done in GA (i.e., GA diversi-

fies the search too much), since the solutions generated by crossover operators usually do not preserve sufficient amount of structures of the original solutions compared to the neighborhood operators of LS. As for (iii), the ability of intensification of algorithms ILS, GLS and SA (and its variants), all of which are based on the mechanism of searching around good solutions by using LS, is higher than other tested metaheuristic algorithms.

5.2 Neighborhoods

The neighborhood (II-A in Section 3) is one of the most important factors in designing LS algorithms. It should be designed so that the structures common to good solutions are preserved after the neighborhood operation is applied, as discussed in Section 5.1. For TSP, for example, the insertion and swap neighborhoods are applicable, which were used in the experiments on SMP in Section 4.2, as TSP is also a sequencing problem. Other common neighborhoods are *Or-opt* and λ-*opt* neighborhoods. Or-opt neighborhood is the set of solutions obtainable by deleting a sub-path of length at most three, and inserting it into another position. λ-opt neighborhood is the set of solutions obtainable by changing at most λ edges of the tour, where λ (≥ 2) is a given constant. Among these, λ-opt neighborhood with $\lambda = 2, 3$ and Or-opt neighborhood are known to be effective, and swap neighborhood is not competitive with others [98, 87]. This is mainly because λ-opt and Or-opt neighborhoods are defined on tour edges, which are tied directly with the objective function of TSP. In many cases, more than one definition of the neighborhood is possible and the performance of the resulting algorithm is strongly affected by the adopted neighborhood. If there is more than one promising neighborhood, they can be combined to make more powerful algorithms [163, 164, 165].

There is a sophisticated mechanism called *variable depth search*, which is an embedded neighborhood construction that compounds simple moves to create more complex and powerful moves. This approach was first successfully applied to the graph partitioning problem by Kernighan and Lin [90], and then to TSP by Lin and Kernighan [102]. Similar but slightly more general idea called the *ejection chain* is applied to various problems under the framework of tabu search [21, 58, 127, 163]. But the neighborhood size becomes exponentially large if we use these ideas without restriction. Therefore, it is necessary to carefully choose the basic simple moves, which are then used to create complex moves, in such a way that the problem structure is exploited as much as possible. For details see, e.g., the original papers by Lin and Kernighan [90, 102], Chapter 19 of [126] or [87]. See also recent applications of the variable depth search such as [21, 127, 164, 165, 163]. Note that a similar idea is also used in [13] and is called the guided local search; however, this is different from the one explained in Section 3.2.

It is also important to consider how to search the neighborhood efficiently. For this purpose, it is often effective to eliminate from consideration those solutions having no possibility of improvement. Here we show a simple example for the 2-opt neighborhood of TSP. Assume that a better solution is found by deleting two edges $\{i,j\}$ and $\{k,l\}$ from the current tour and by adding $\{i,k\}$ and $\{j,l\}$; i.e., $d_{ij} + d_{kl} > d_{ik} + d_{jl}$. This implies that at least one of the conditions $d_{ij} > d_{jl}$ or $d_{kl} > d_{ik}$ holds. Hence, while considering the 2-opt moves in which edge $\{i,j\}$ is deleted, we can limit the candidate edges to add to those $\{j,l\}$ with $d_{jl} < d_{ij}$ or those $\{i,k\}$ with $d_{ik} < d_{ij}$. Note that if we decide to delete edge $\{i,j\}$ and to add edge $\{j,l\}$, then the remaining edges $\{k,l\}$ to be deleted and $\{i,k\}$ to be added are uniquely determined to maintain feasibility. A similar reduction is also possible for the 3-opt neighborhood. Lin–Kernighan neighborhood can be considered as a generalization of this idea to the λ-opt neighborhood with general λ, although it is not guaranteed to reach locally optimal solutions. For Euclidean TSP, in which the coordinates of the cities are given in the k-dimensional space (k (≥ 2) is a constant), and the distances are the Euclidean distances between them, a special data structure called K-d tree can be used to enumerate all cities l satisfying $d_{jl} < d_{ij}$ for a given edge $\{i,j\}$ efficiently [16]. Data structures for executing basic neighborhood operations efficiently are also extensively studied; e.g., [48]. Other techniques to improve the efficiency of local search for TSP can be found in [87].

Similar reductions of neighborhoods as above are also possible for MAXSAT. We proposed in [160, 161] an efficient data structure to reduce the sizes of the 2- and 3-flip neighborhoods (i.e., N_2 and N_3). We show that the expected size of the reduced 2-flip neighborhood is $O(n+m)$, assuming randomly generated solutions, while the original size of N_2 is $O(n^2)$. It is also observed by the computational experiments that the expected size represents the real performance of the local search well. Similar results are also reported for the 3-flip neighborhood.

5.3 Other techniques

Metaheuristic algorithms usually include program parameters to control the search, some of which must be carefully tuned to obtain sufficient performance. It must be tedious if such tuning is needed whenever the size and/or the type of instances change. To avoid this, important parameters are sometimes adaptively controlled. Among the targets of such adaptive control are the tabu tenure of TS (parameter t_{tabu} in Sections 4.2 and 4.5) [14, 116] and the penalty weights of the penalty function method (Section 3.2) [52, 163].

If the problem can be naturally formulated as an integer programming problem, it is often useful to exploit the information from its Lagrangian or linear programming relaxation problem; e.g., the relative costs, the optimal solution to the relaxed problem and so on. Although there are many early references on such approaches (e.g. see the

survey [15] by Beasley), more improvement is still expected, as can be understood by the outstanding results reported recently for the set covering problem [22, 23]. This type of approaches are mainly based on one of the following three ideas.

- Reduce the search space by fixing some variables by using the information of relative costs.
- Generate feasible solutions of the original problem by modifying optimal (or sufficiently good) solutions to the relaxed problem.
- Generate feasible solutions by using greedy methods in which the relative costs are used in the local evaluation of elements.

It is also possible to reduce the neighborhood size by using the information of the relative costs [163]. The information from the surrogate dual can also be used to control metaheuristic algorithms (e.g., TS in [71]). The use of the information of the relaxation problems may not be limited to these, especially if it is combined with metaheuristic algorithms.

If exact or heuristic algorithms are available for partial problems of the original problem, they can be used as basic operators of metaheuristics. For example, in [158], a genetic crossover operator is proposed in which an optimal sequence among those having no contradiction with the partial order defined by two parent solutions is found by using *dynamic programming* (DP). A similar idea is also applied to the independent set problem and is called the *optimized crossover* [5]. See [80, 133] for details of such hybrid approaches.

From the view point of practical use, it would be convenient if we have a good metaheuristic algorithm for a general problem into which many problems can be formulated. Such an algorithm may be called a general problem solver. If a general problem solver to problem A is available, it would be easier to formulate the problem we want to solve as an instance of A and then to apply the general problem solver, than to develop a new algorithm for the particular problem from scratch. Such metaheuristic algorithms have already been developed. For example, a tabu search algorithm for the constraint satisfaction problem (abbreviated as CSP) was proposed as a general problem solver in [116, 118], which was then used to solve many real-world problems (by reducing them to CSP) and was proved to be quite effective. Another is a tabu search for 0-1 mixed integer programming problem [103]. In [155], the idea of WALKSAT (Section 3.3) for problem SAT was generalized to solve more general integer programming problem. A general problem solver for the resource constrained project scheduling problem was developed in [117].

6 Theoretical results

In this section, we describe theoretical aspects of metaheuristic algorithms such as asymptotic convergence of SA, computational complexity of simple LS and approximation ratio of locally optimal solutions.

SA is known to converge to an exact optimal solution asymptotically under appropriate assumptions [93, 106], which is considered to be one of the attractive features of SA. Similar analyses exist for TA, GA and probabilistic TS, in which probabilistic move strategy is incorporated in TS [7, 42, 141]. However, these analytical results do not directly relate to their real performance, since such analyses assume "sufficiently long" computational time, which is not realistic. Actually, negative results are known on the computational time needed for SA to converge. For example, it is shown in [140] that SA requires exponential time to find an optimal solution for the matching problem, for which polynomial time algorithms exist. In view of these, it might be better not to stick to the analytical results or analogy to the physical phenomena of annealing in designing metaheuristic algorithms such as SA. Flexible thinking is more important.

Neighborhood $N(x)$ of simple LS is usually designed so that the computational time needed for one move is polynomially bounded; however, the number of moves to reach a locally optimal solution from an initial solution may not be polynomial. (However, we can bound the number of such moves by the number of different values $\tilde{f}$ can take, which is usually pseudo-polynomial, since the value of $\tilde{f}$ of the candidate solution is decreased at least by one in every move.) In fact, there exist instances and initial solutions of TSP such that LS with 2-opt neighborhood will require $\Theta(2^{n/2})$ moves before halting. A similar result is also shown for the 3-opt [25], and this type of results for TSP are summarized in [87]. To analyze the complexity of finding a locally optimal solution, Johnson et al. [88] introduced a complexity class called *PLS* (*polynomial-time local search*). Each problem in class PLS is defined as a pair of an optimization problem and the definition of neighborhood. Many important local search problems are known to be complete for class PLS, which means that the time to find a locally optimal solution by any method can not be polynomially bounded unless all PLS problems (to find locally optimal solutions) can be solved in polynomial time. For example, the local search problem for TSP under λ-opt neighborhood for a sufficiently large constant λ and that for MAXSAT under 1-flip neighborhood are known to be PLS-complete. See the extensive survey of Yannakakis [168] for the theory of PLS-completeness.

Contrary to these theoretical results, the time to reach a locally optimal solution is usually fast in practice. To investigate this direction, there are attempts to obtain theoretical bounds on the average computation time to find a locally optimal solution under appropriate probabilistic models. For example, the average number of moves

that the local search algorithm with 2-opt neighborhood will make is bounded by $O(n^{10} \log n)$ for randomly generated two-dimensional Euclidean TSP instances [25]. This analysis is generalized to λ-opt case in [119]. However, it is reported in the experimental results by Johnson and McGeoch [87] that the average number of moves is approximately $\Theta(n \log n)$, which is much smaller than the above theoretical bound.

There are also theoretical results on the quality of locally optimal solutions. Let x^* be an optimal solution and x be a solution obtained by an approximate algorithm. Then a bound on the ratio $f(x)/f(x^*)$ is called an *approximation ratio*. For simplicity, $f(x) > 0$ is assumed for all $x \in F$. Then the approximation ratio is not less (resp., more) than 1 for minimization (resp., maximization) problems, which is considered better if it is closer to 1. For example, it is known for TSP that the approximation ratio of locally optimal solutions will not be bounded by a constant unless P = NP, if the local optimality can be checked in polynomial time [125]. (See [51] for classes P and NP.) It is also known for TSP that no polynomial time algorithm can achieve approximation ratio $2^{poly(n)}$ unless P = NP, where $poly(n)$ is any fixed polynomial on n [139]; however, this result can not be directly applied to LS, since the time complexity of LS is not always polynomially bounded. The situation becomes better if some restrictions are imposed on the instances, like the cases discussed in [25] (see also [87]). If the distance matrix (d_{ij}) satisfies the triangle inequality, i.e., $d_{ij} \leq d_{ik} + d_{kj}$ holds for all i, j and k, the approximation ratio is at most $4\sqrt{n}$ for 2-opt, while it is also shown that the ratio is at least $(1/4)\sqrt{n}$ for 2-opt and is at least $(1/4)n^{1/(2\lambda)}$ for λ-opt for any $\lambda \geq 3$. For Euclidean TSP in the k-dimensional space, where k is a fixed constant, the approximation ratio is at most $O(\log n)$ for 2-opt, while it is at least $\Omega(\log n / \log \log n)$ even if $k = 2$. Moreover, if the coordinates of cities are randomly chosen in the k-dimensional space, the expected approximation ratio of the locally optimal tour with respect to 2-opt is bounded by a constant. However, much better approximation ratio are achieved by polynomial time algorithms which are not based on LS. The solutions obtained by the algorithm of Christofides have approximation ratio of 1.5 if the distance matrix satisfies the triangle inequality [28]. Arora proposed a *PTAS* (*polynomial time approximation scheme*) for Euclidean TSP, i.e., the obtained solution will not exceed $1 + \varepsilon$ times the optimal for arbitrary $\varepsilon > 0$ [8] (see also the tutorial in [148]). From these results, LS appears to be inferior to such algorithms from the theoretical point of view.

On the other hand, the computational time of the PTAS by Arora is $O(n((\log n)/\varepsilon)^{100/\varepsilon})$ [148], which is polynomial but is not practical. Moreover, it was observed in the computational experiments in [87] that the average approximation ratios for randomly generated 2-dimensional Euclidean TSP instances with $n = 1000$ were 1.097 for Christofides, 1.079 for the 2-opt and 1.038 for the 3-opt, which are much better than the theoretical bounds.

Similar analyses are also known for MAXSAT. It is shown in [72] that the approximation ratio of locally optimal solutions with respect to the 1-flip neighborhood N_1 is $q/(q+1)$ if the number of literals in one clause is not less than q. It is also shown that this bound is tight, i.e., best possible. (Although the proof in [72] is limited to the case with $w_i = 1$ for all i, it is easily generalized to the case with general weights. Moreover, although the tightness result in [72] was shown by giving an example of a fixed size and a fixed q, it is easily generalized to the case with general size and q.)

For MAXSAT, however, much better results are known. These are attained from the analyses of the expected objective values, assuming that the 0-1 assignment of each variable is randomly chosen. Among the known approximation ratios of this type are 0.5 by Johnson [84] (Chen et al. later show that the approximation ratio of Johnson's algorithm is better than 0.5, and is 2/3 [27]), 0.75 by Yannakakis [167], 0.75 [65] and 0.7584 [66] by Goemans and Williamson and so on. Much effort has been devoted to improving the approximation ratio, and one of the latest results is 0.7846 by Asano and Williamson [10]. The algorithms proposed by Goemans and Williamson were based on the linear programming (LP) and semidefinite programming (SDP) relaxations, which received much attention recently as general tools to get good approximation ratios. See the tutorial [9] for details. The above results are general in the sense they do not involve parameter q in the results; however, if q is used, approximation ratio of $1 - 1/2^q$ is attained by Johnson's algorithm. Compared to these, the approximation ratio $q/(q+1)$ of LS with N_1 is inferior.

On the other hand, from the practical point of view, the following results were observed in the computational results on randomly generated instances of Section 4.4. (While the results in Section 4.6 were for 17 instances with $m = 850$, the results here are the average of 44 instances in which those instances with $m = 800$ and 900 are also included.) Asano conducted computational experiments on Johnson's algorithm, the LP-based algorithm by Goemans and Williamson, and some variants of them in [9]. The average approximation ratios of randomly generated solutions by assigning 1 to each variable by the probability given by the algorithm were about 0.935 for Johnson's algorithm and 0.95 for the best one among the LP relaxation based methods. (Note that the theoretical bounds on approximation ratios are based on the expected objective values of such random solutions.) The average approximation ratios of the solutions obtained by the greedy methods whose local evaluation are defined on the basis of the probabilistic analyses were about 0.997 for Johnson's algorithm and 0.998 for the best one among LP relaxation based methods, which were much better than the randomly generated solutions. We also conducted computational experiments on simple LS algorithms with different neighborhoods, N_1, N_2 and N_3. The average approximation ratios were about 0.9927 for N_1, 0.9970 for N_2 and 0.9982 for N_3. We also tested MLS in which the LS was repeated 1000 times. Then the average approximation

ratios were improved to 0.9988 for N_1 and 0.99988 for N_2, where their computational time for each instance were about 3.9 seconds for N_1 and 16.8 seconds for N_2 on a workstation Sun Ultra 2 Model 2300. These results indicate that the real performance of heuristic algorithms are much better than the worst case theoretical bounds, and much has to be done to reduce the gap between them.

There are many other attempts to analyze metaheuristic algorithms theoretically. For example, there are results of probabilistic analyses on GA such as schema theorem [11, 68]. Another interesting result is the probabilistic analysis of an algorithm called *go with the winners*, which is a population based heuristic similar to GA. It is shown, under an appropriate (but rather strict) assumptions, that the go with the winners algorithm performs better than simple LS algorithms [6, 35]. However, much of theoretical aspects of metaheuristics has yet been understood. Especially it is not known theoretically why metaheuristics work well in practice. Therefore, much has to be done in this direction.

7 Conclusion

In this paper, we surveyed metaheuristic algorithms for combinatorial optimization problems, and tried to explain the underlying ideas systematically based on a generalized framework of local search. We then conducted computational experiments on the problems of single machine scheduling and maximum satisfiability to understand general performances of metaheuristics. Various metaheuristic algorithms were compared from the view point of simple optimization tools. Simple guidelines on how to design metaheuristic algorithms are derived from these computational results. We also surveyed advanced techniques which have been proposed to improve the performance of metaheuristics. Some theoretical results on metaheuristics were also mentioned.

Recently, metaheuristics are widely acknowledged as one of the most effective tools to solve large-scale instances of combinatorial optimization problems. However, not much of its theoretical aspects has been understood, and innovative ideas would be necessary to give it a breakthrough. We hope that this article will be of some help for the future research on metaheuristics.

Acknowledgment

The authors are grateful to anonymous referees for their valuable comments. This research was partially supported by Scientific Grant-in-Aid, by the Ministry of Education, Science, Sports and Culture of Japan.

References

[1] E.H.L. Aarts and J.H.M. Korst, *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons, 1989.

[2] E.H.L. Aarts and J.K. Lenstra, eds., *Local Search in Combinatorial Optimization*, John Wiley & Sons, 1997.

[3] E.H.L. Aarts, J.H.M. Korst and P.J.M van Laarhoven, "Simulated annealing," in: E.H.L. Aarts and J.K. Lenstra, eds., *Local Search in Combinatorial Optimization*, John Wiley & Sons, pp.91–120, 1997.

[4] E.H.L. Aarts, P.J.M. van Laarhoven, J.K. Lenstra and N.L.J. Ulder, "A computational study of local search algorithms for job shop scheduling," *ORSA J. Computing*, vol.6, pp.118–125, 1994.

[5] C.C. Aggarwal, J.B. Orlin and R.P. Tai, "Optimized crossover for the independent set problem," *Operations Research*, vol.45, pp.226–234, 1997.

[6] D. Aldous and U. Vazirani, ""Go with the winners" algorithms," *Proc. 35th Annual Symposium on Foundations of Computer Science*, pp.492–501, 1994.

[7] I. Althöfer and K.-U. Koschnick, "On the convergence of "threshold accepting"," *Applied Mathematics and Optimization*, vol.24, pp.183–195, 1991.

[8] S. Arora, "Nearly linear time approximation schemes for Euclidean TSP and other geometric problems," *Proc. 38th Annual Symposium on Foundations of Computer Science*, pp.554–563, 1997.

[9] T. Asano, "High performance approximation algorithms for logical system analysis (in Japanese)," *IPSJ Magazine*, vol.39, pp.683–688, 1998.

[10] T. Asano and D.P. Williamson, "Simpler approximation algorithms for MAX SAT," Technical Report of IEICE, COMP99-14, pp.33–40, 1999.

[11] T. Bäck, D.B. Fogel and Z. Michalewicz, eds., *Handbook of Evolutionary Computation*, IOP Publishing Ltd and Oxford University Press, 1997.

[12] K.R. Baker and G.D. Scudder, "Sequencing with earliness and tardiness penalties: a review," *Operations Research*, vol.38, pp.22–36, 1990.

[13] E. Balas and A. Vazacopoulos, "Guided local search with shifting bottleneck for job shop scheduling," *Management Science*, vol.44, pp.262–275, 1998.

[14] R. Battiti and G. Tecchiolli, "Local search with memory: benchmarking RTS," *OR Spektrum*, vol.17, pp.67–86, 1995.

[15] J.E. Beasley, "Lagrangian heuristics for location problems," *European J. Operational Research*, vol.65, pp.383–399, 1993.

[16] J.J. Bentley, "Fast algorithms for geometric traveling salesman problems," *ORSA J. Computing*, vol.4, pp.387–410, 1992.

[17] K.D. Boese, A.B. Kahng and S. Muddu, "A new adaptive multi-start technique for combinatorial global optimizations," *Operations Research Letters*, vol.16, pp.101–

113, 1994.

[18] R.M. Brady, "Optimization strategies gleaned from biological evolution," *Nature*, vol.317, pp.804–806, 1985.

[19] J. Brimberg, P. Hansen, N. Mladenović and É.D. Taillard, "Improvements and comparison of heuristics for solving the uncapacitated multisource weber problem," *Operations Research*, vol.48, pp.444–460, 2000.

[20] J. Brimberg and N. Mladenović, "A variable neighborhood algorithm for solving the continuous location-allocation problem," *Studies in Locational Analysis*, vol.10, pp.1–12, 1996.

[21] B. Cao and F. Glover, "Tabu search and ejection chains — application to a node weighted version of the cardinality-constrained TSP," *Management Science*, vol.43, pp.908–921, 1997.

[22] A. Caprara, M. Fischetti and P. Toth, "A heuristic method for the set covering problem," *Operations Research*, vol.47, pp.730–743, 1999.

[23] S. Ceria, P. Nobili and A. Sassano, "A Lagrangian-based heuristic for large-scale set covering problems," *Mathematical Programming*, vol.81, pp.215–228, 1998.

[24] V. Černý, "Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm," *J. Optimization Theory and Applications*, vol. 45, pp.41–51, 1985.

[25] B. Chandra, H. Karloff and C.A. Tovey, "New results on the old k-opt algorithm for the traveling salesman problem," *SIAM J. Computing*, vol.28, pp.1998–2029, 1999.

[26] I. Charon and O. Hudry, "The noising method: a new method for combinatorial optimization," *Operations Research Letters*, vol.14, pp.133–137, 1993.

[27] J. Chen, D.K. Friesen and H. Zheng, "Tight bound on Johnson's algorithm for maximum satisfiability," *J. Computer and System Sciences*, vol.58, pp.622–640, 1999.

[28] N. Christofides, *Worst-case analysis of a new heuristic for the traveling salesman problem*, Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA, 1976.

[29] B. Codenotti, G. Manzini, L. Margara and G. Resta, "Perturbation: an efficient technique for the solution of very large instances of the Euclidean TSP," *INFORMS J. Computing*, vol.8, pp.125–133, 1996.

[30] A. Colorni, M. Dorigo and V. Maniezzo, "Distributed optimization by ant colonies, " *Proc. European Conference on Artificial Life*, pp.134–142, 1991.

[31] H.A.J. Crauwels, C.N. Potts and L.N. van Wassenhove, "Local search heuristics for single-machine scheduling with batching to minimize the number of late jobs," *European J. Operational Research*, vol.90, pp.200–213, 1996.

[32] G.A. Croes, "A method for solving traveling-salesman problems," *Operations Re-*

search, vol.6, pp.791–812, 1958.

[33] L. Davis, "Applying adaptive algorithms to epistatic domains," *Proc. 9th International Joint Conference on Artificial Intelligence*, pp.162–164, 1985.

[34] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.

[35] T. Dimitriou and R. Impagliazzo, "Towards an analysis of local optimization algorithms," *Proc. 28th Annual ACM Symposium on the Theory of Computing*, pp.304–313, 1996.

[36] M. Dorigo, *Optimization, Learning, and Natural Algorithms*, PhD Thesis, Politecnico di Milano, 1992.

[37] M. Dorigo and L.M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol.1, pp.53–66, 1997.

[38] M. Dorigo, V. Maniezzo and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics — Part B*, vol.26, pp.29–41, 1996.

[39] J. Dorn, M. Girsch, G. Skele and W. Slany, "Comparison of iterative improvement techniques for schedule optimization," *European J. Operational Research*, vol.94, pp.349–361, 1996.

[40] G. Dueck, "New optimization heuristics: the great deluge algorithm and the record-to-record travel," *J. Computational Physics*, vol.104, pp.86–92, 1993.

[41] G. Dueck and T. Scheuer, "Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing," *J. Computational Physics*, vol.90, pp.161–175, 1990.

[42] U. Faigle and W. Kern, "Some convergence results for probabilistic tabu search," *ORSA J. Computing*, vol.4, pp.32–37, 1992.

[43] T.A. Feo and J.L. González-Velarde, "The intermodal trailer assignment problem," *Transportation Science*, vol.29, pp.330–341, 1995.

[44] T.A. Feo and M.G.C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol.8, pp.67–71, 1989.

[45] T.A. Feo and M.G.C. Resende, "Greedy randomized adaptive search procedures," *J. Global Optimization*, vol.6, pp.109–133, 1995.

[46] T.A. Feo, M.G.C. Resende and S.H. Smith, "A greedy randomized adaptive search procedure for maximum independent set," *Operations Research*, vol.42, pp.860–878, 1994.

[47] T.A. Feo, K. Venkatraman and J.F. Bard, "A GRASP for a difficult single machine scheduling problem," *Computers and Operations Research*, vol.18, pp.635–643, 1991.

[48] M.L. Fredman, D.S. Johnson, L.A. McGeoch and G. Ostheimer, "Data structures for traveling salesmen," *J. Algorithms*, vol.18, pp.432–479, 1995.

[49] B. Freisleben and P. Merz, "A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems," *Proc. IEEE International Conference on Evolutionary Computation*, pp.616–621, 1996.

[50] L.M. Gambardella, É.D. Taillard and M. Dorigo, "Ant colonies for the quadratic assignment problem," *J. Operations Research Society*, vol.50, pp.167–176, 1999.

[51] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[52] M. Gendreau, A. Hertz and G. Laporte, "A tabu search heuristic for the vehicle routing problem," *Management Science*, vol.40, pp.1276–1290, 1994.

[53] F. Glover, "Heuristics for integer programming using surrogate constraints," *Decision Sciences*, vol.8, pp.156–166, 1977.

[54] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers and Operations Research*, vol.13, pp.533–549, 1986.

[55] F. Glover, "Tabu search — part I," *ORSA J. Computing*, vol.1, pp.190–206, 1989; part II, ditto, vol.2, pp.4–32, 1990.

[56] F. Glover, "Genetic algorithms and scatter search: unsuspected potentials," *Statistics and Computing*, vol.4, pp.131–140, 1994.

[57] F. Glover, "Scatter search and star-paths: beyond the genetic metaphor," *OR Spektrum*, vol.17, pp.125–137, 1995.

[58] F. Glover, "Ejection chains, reference structures and alternating path methods for traveling salesman problems," *Discrete Applied Mathematics*, vol.65, pp.223–253, 1996.

[59] F. Glover, "Tabu search and adaptive memory programming — advances, applications and challenges," in: R.S. Barr, R.V. Helgason and J.L. Kennington, eds., *Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies*, Kluwer Academic Publishers, 1997.

[60] F. Glover and G.A. Kochenberger, "Critical events tabu search for multidimensional knapsack problem," in: I.H. Osman and J.P. Kelly, eds., *Meta-Heuristics: The Theory and Applications*, Kluwer Academic Publishers, pp.407–427, 1996.

[61] F. Glover, G.A. Kochenberger and B. Alidaee, "Adaptive memory tabu search for binary quadratic problems," *Management Science*, vol.44, pp.336–345, 1998.

[62] F. Glover and M. Laguna, "Tabu search," in: *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publishing, pp.70–141, 1993.

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

[64] F. Glover, É.D. Taillard and D. de Werra, "A user's guide to tabu search," *Annals of Operations Research*, vol.41, pp.3–28, 1993.

[65] M.X. Goemans and D.P. Williamson, "New 3/4-approximation algorithms for the maximum satisfiability problem," *SIAM J. Discrete Mathematics*, vol.7, pp.656–

666, 1994.

[66] M.X. Goemans and D.P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. Association for Computing Machinery*, vol.42, pp.1115–1145, 1995.

[67] D.E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," *Proc. 1st International Conference on Genetic Algorithms*, pp.154–159, 1985.

[68] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[69] J. Gu, "Optimization by multispace search," Technical Report UCECE-TR-90-001, Department of Electrical and Computer Engineering, University of Calgary, 1990.

[70] J. Gu and X. Huang, "Efficient local search with search space smoothing: a case study of the traveling salesman problem (TSP)," *IEEE Transactions on Systems, Man, and Cybernetics*, vol.24, pp.728–735, 1994.

[71] S. Hanafi and A. Freville, "An efficient tabu search approach for the 0-1 multidimensional knapsack problem," *European J. Operational Research*, vol.106, pp.659–675, 1998.

[72] P. Hansen and B. Jaumard, "Algorithms for the maximum satisfiability problem," *Computing*, vol.44, pp.279–303, 1990.

[73] P. Hansen and N. Mladenović, "An introduction to variable neighborhood search," in: S. Voß, S. Martello, I.H. Osman and C. Roucairol, eds., *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, pp.433–458, 1999.

[74] J. Hart and A. Shogan, "Semi-greedy heuristics: an empirical study," *Operations Research Letters*, vol.6, pp.107–114, 1987.

[75] A. Hertz, É.D. Taillard and D. de Werra, "Tabu search," in: E.H.L. Aarts and J.K. Lenstra, eds., *Local Search in Combinatorial Optimization*, John Wiley & Sons, pp.121–136, 1997.

[76] J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, The University of Michigan Press, 1975, and MIT Press, 1992.

[77] J.N. Hooker and C. Fedjki, "Branch-and-cut solution of inference problems in propositional logic," *Annals of Mathematics and Artificial Intelligence*, vol.1, pp.123–139, 1990.

[78] T. Ibaraki, "Methods of combinatorial optimization — from the example of the traveling salesman problem — (in Japanese)," *The Transactions of the Institute of Electrical Engineers of Japan*, vol.114-C, pp.411–419, 1994.

[79] T. Ibaraki, "Combinatorial optimization and scheduling problems: new algorithms and their perspective (in Japanese)," *Journal of the Society of Instrument and*

Control Engineers, vol.34, pp.340–346, 1995.

[80] T. Ibaraki, "Combination with other optimization methods," in: T. Bäck, D.B. Fogel and Z. Michalewicz, eds., *Handbook of Evolutionary Computation*, IOP Publishing Ltd and Oxford University Press, 1997.

[81] T. Ibaraki and Y. Nakamura, "A dynamic programming method for single machine scheduling," *European J. Operational Research*, vol.76, pp.72–82, 1994.

[82] P. Jog, J.Y. Suh and D. van Gucht, "The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem," *Proc. 3rd International Conference on Genetic Algorithms*, pp.110–115, 1989.

[83] P. Jog, J.Y. Suh and D. van Gucht, "Parallel genetic algorithms applied to the traveling salesman problem," *SIAM J. Optimization*, vol.1, pp.515–529, 1991.

[84] D.S. Johnson, "Approximation algorithms for combinatorial problems," *J. Computer and System Sciences*, vol.9, pp.256–278, 1974.

[85] D.S. Johnson, "Local optimization and the traveling salesman problem," *Proc. 17th Colloquium on Automata, Languages and Programming*, pp.446–461, 1990.

[86] D.S. Johnson, C.R. Aragon, L.A. McGeoch and C. Schevon, "Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning," *Operations Research*, vol.37, pp.865–892, 1989; "part II, graph coloring and number partitioning," ditto, vol.39, pp.378–406, 1991.

[87] D.S. Johnson and L.A. McGeoch, "The traveling salesman problem: a case study," in: E.H.L. Aarts and J.K. Lenstra, eds., *Local Search in Combinatorial Optimization*, John Wiley & Sons, pp.215–310, 1997.

[88] D.S. Johnson, C.H. Papadimitriou and M. Yannakakis, "How easy is local search?, " *J. Computer and System Sciences*, vol.37, pp.79–100, 1988.

[89] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for GAs," *Proc. 6th International Conference on Genetic Algorithms*, pp.110–115, 1995.

[90] B.W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Technical J.*, vol.49, pp.291–307, 1970.

[91] P. Kilby, P. Prosser and P. Shaw, "Guided local search for the vehicle routing problem with time windows," in: S. Voß, S. Martello, I.H. Osman and C. Roucairol, eds., *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, pp.473–486, 1999.

[92] S. Kirkpatrick, C.D. Gelatt, Jr. and M.P. Vecchi, "Optimization by simulated annealing," *Science*, vol.220, pp.671–680, 1983.

[93] H. Kise, "Simulated annealing for scheduling problems (in Japanese)," *Communications of the Operations Research Society of Japan*, vol.40, pp.268–273, 1995.

[94] H. Kitano, ed., *Genetic Algorithms* (in Japanese), Sangyo-Tosho, 1993; vol.2, ditto,

1995; vol.3, ditto, 1997.

[95] A. Kolen and E. Pesch, "Genetic local search in combinatorial optimization," *Discrete Applied Mathematics*, vol.48, pp.273–284, 1994.

[96] M. Kubo, "Metaheuristics (in Japanese)," in: *Discrete Structures and Algorithms*, vol.IV, Kindai-Kagaku-Sha, pp.171–230, 1995.

[97] M. Laguna, T.A. Feo and H.C. Elrod, "A greedy randomized adaptive search procedure for the two-partition problem," *Operations Research*, vol.42, pp.677–687, 1994.

[98] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, eds., *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, 1985.

[99] Y. Li, P.M. Pardalos and M.G.C. Resende, "A greedy randomized adaptive search procedure for the quadratic assignment problem," *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, vol.16, pp.237–261, 1994.

[100] C.K.Y. Lin, K.B. Haley and C. Sparks, "A comparative study of both standard and adaptive versions of threshold accepting and simulated annealing algorithms in three scheduling problems," *European J. Operational Research*, vol.83, pp.330–346, 1995.

[101] S. Lin, "Computer solutions of the traveling salesman problem," *Bell System Technical J.*, vol.44, pp.2245–2269, 1965.

[102] S. Lin and B.W. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," *Operations Research*, vol.21, pp.498–516, 1973.

[103] A. Løkketangen and F. Glover, "Solving zero-one mixed integer programming problems using tabu search," *European J. Operational Research*, vol.106, pp.624–658, 1998.

[104] H.R. Lourenço, "Job-shop scheduling: computational study of local search and large-step optimization methods," *European J. Operational Research*, vol.83, pp.347–364, 1995.

[105] H.R. Lourenço and D. Serra, "Adaptive approach heuristics for the generalized assignment problem," Technical Report: Department of Economics and Management, Univerisitat Pompeu Fabra, R. Trias Fargas 25-27, 08005 Barcelona, Spain, 1998 (available at `http://www.econ.upf.es/deehome/what/wpapers/listwork.html`).

[106] M. Lundy and A. Mees, "Convergence of an annealing algorithm," *Mathematical Programming*, vol.34, pp.111–124, 1986.

[107] B. Manderick, M. de Weger and P. Spiessens, "The genetic algorithm and the structure of the fitness landscape," *Proc. 4th International Conference on Genetic Algorithms*, pp.143–150, 1991.

[108] O.C. Martin and S.W. Otto, "Combining simulated annealing with local search

heuristic," *Annals of Operations Research*, vol.63, pp.57–75, 1996.

[109] O. Martin, S.W. Otto and E.W. Felten, "Large-step Markov chains for the traveling salesman problem," *Complex Systems*, vol.5, pp.299–326, 1991.

[110] O. Martin, S.W. Otto and E.W. Felten, "Large-step Markov chains for the TSP incorporating local search heuristic," *Operations Research Letters*, vol.11, pp.219–224, 1992.

[111] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1992.

[112] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers and Operations Research*, vol.24, pp.1097–1100, 1997.

[113] H. Mühlenbein, "Parallel genetic algorithms, population genetics and combinatorial optimization," *Proc. 3rd International Conference on Genetic Algorithms*, pp.416–421, 1989.

[114] H. Mühlenbein, M. Gorges-Schleuter and O. Krämer, "Evolution algorithms in combinatorial optimization," *Parallel Computing*, vol.7, pp.65–85, 1988.

[115] H. Mühlenbein, "Genetic algorithms," in: E.H.L. Aarts and J.K. Lenstra, eds., *Local Search in Combinatorial Optimization*, John Wiley & Sons, pp.137–171, 1997.

[116] K. Nonobe and T. Ibaraki, "A tabu search approach to the constraint satisfaction problem as a general problem solver," *European J. Operational Research*, vol.106, pp.599–623, 1998.

[117] K. Nonobe and T. Ibaraki, "Formulation and tabu search algorithm for the resource constrained project scheduling problem (RCPSP)," Technical Report #99010, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, 1999.

[118] K. Nonobe and T. Ibaraki, "An improved tabu search method for the weighted constraint satisfaction problem," Technical Report #99022, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, 1999; *INFOR* (to appear).

[119] M. Okada, *Studies on Probabilistic Analysis of λ-opt for Traveling Salesperson Problems*, Doctoral dissertation, NAIST-IS-DT9661004, Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan, 1999.

[120] I.M. Oliver, D.J. Smith and J.R.C. Holland, "A study of permutation crossover operators on the traveling salesman problem," *Proc. 2nd International Conference on Genetic Algorithms*, pp.224–230, 1987.

[121] I.H. Osman and J.H. Kelly, "Meta-heuristics: an overview," in: I.H. Osman and J.P. Kelly, eds., *Meta-Heuristics: Theory and Applications*, Kluwer Academic Publishers, pp.1–21, 1996.

[122] I.H. Osman and J.P. Kelly, eds., *Meta-Heuristics: Theory and Applications*,

Kluwer Academic Publishers, 1996.

[123] I.H. Osman and G. Laporte, "Metaheuristics: a bibliography," in: G. Laporte and I.H. Osman, eds., *Metaheuristics in Combinatorial Optimization*, Annals of Operations Research 63, Baltzer, pp.513–623, 1996.

[124] C.H. Papadimitriou, "On selecting a satisfying truth assignment (extended abstract)," *Proc. 32nd Annual Symposium on Foundations of Computer Science*, pp.163–169, 1991.

[125] C.H. Papadimitriou and K. Steiglitz, "On the complexity of local search for the traveling salesman problem," *SIAM J. Computing*, vol.6, pp.76-83, 1977.

[126] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, 1982; re-issued by Dover Publications, 1998.

[127] E. Pesch and F. Glover, "TSP ejection chains," *Discrete Applied Mathematics*, vol.76, pp.165-181, 1997.

[128] E. Pesch and S. Voß, "Strategies with memories: local search in an application oriented environment," *OR Spektrum*, vol.17, pp.55–66, 1995.

[129] M. Pirlot, "General local search heuristics in combinatorial optimization: a tutorial," *Belgian J. Operations Research, Statistics and Computer Science*, vol.32, pp.7-67, 1992.

[130] M. Pirlot, "General local search methods," *European J. Operational Research*, vol.92, pp.493–511, 1996.

[131] V.J. Rayward-Smith, I.H. Osman, C.R. Reeves and G.D. Smith, eds., *Modern Heuristic Search Methods*, John Wiley & Sons, 1996.

[132] C.R. Reeves, ed., *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publications, 1993; re-issued by McGraw-Hill, 1995.

[133] C.R. Reeves, "Genetic algorithms for the operations researcher," *INFORMS J. Computing*, vol.9, pp.231–250, 1997.

[134] C.R. Reeves, "Landscapes, operators and heuristic search," *Annals of Operations Research*, vol.86, pp.473–490, 1999.

[135] S. Reiter and D.B. Rice, "Discrete optimizing solution procedures for linear and nonlinear integer programming problems," *Management Science*, vol.12, pp.829–850, 1966.

[136] S. Reiter and G. Sherman, "Discrete optimizing," *J. Society for Industrial and Applied Mathematics*, vol.13, pp.864–889, 1965.

[137] M.G.C. Resende and T.A. Feo, "A GRASP for satisfiability," *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, vol.26, pp.499–520, 1996.

[138] M.G.C. Resende, L.S. Pitsoulis and P.M. Pardalos, "Approximate solution of weighted MAX-SAT problems using GRASP," *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, vol.35, pp.393–405, 1997.

[139] S. Sahni and T. Gonzalez, "P-complete approximation problems," *J. Association for Computing Machinery*, vol.23, pp.555–565.

[140] G.H. Sasaki and B. Hajek, "The time complexity of maximum matching by simulated annealing," *J. Association for Computing Machinery*, vol.35, pp.387–403, 1988.

[141] L.M. Schmitt, C.L. Nehaniv and R.H. Fujii, "Linear analysis of genetic algorithms," *Theoretical Computer Science*, vol.200, pp.101–134, 1998.

[142] B. Selman and H.A. Kautz, "Domain-independent extensions to GSAT: solving large structured satisfiability problems," *Proc. 13th International Joint Conference on Artificial Intelligence*, pp.290–295, 1993.

[143] B. Selman, H.A. Kautz and B. Cohen, "Noise strategies for improving local search," *Proc. 12th National Conference on Artificial Intelligence*, pp.337–343, 1994.

[144] M. Sinclair, "Comparison of the performance of modern heuristics for combinatorial optimization on real data," *Computers and Operations Research*, vol.20, pp.687–695, 1993.

[145] T. Stützle and H. Hoos, "$\mathcal{MAX}$-$\mathcal{MIN}$ ant system and local search for the traveling salesman problem," *Proc. 1997 IEEE International Conference on Evolutionary Computation*, 1997.

[146] J.Y. Suh and D. van Gucht, "Incorporating heuristic information into genetic search," *Proc. 2nd International Conference on Genetic Algorithms*, pp.100–107, 1987.

[147] É.D. Taillard, P. Badeau, M. Gendreau, F. Guertin and J.-Y. Potvin, "A tabu search heuristic for the vehicle routing problem with soft time windows," *Transportation Science*, vol.31, pp.170–186, 1997.

[148] H. Tamaki, "Approximation algorithms for the traveling salesman problem: recent major progress by Arora (in Japanese)," *IPSJ Magazine*, vol.39, pp.566–573, 1998.

[149] H. Tamaki, H. Kita, T. Iwamoto and N. Sannomiya, "Genetic algorithms — I — basic concepts (in Japanese)," *Systems, Control and Information*, vol.39, pp.295–302, 1995; H. Tamaki and H. Kita, "II — application to optimization 1 (in Japanese)," ditto, vol.39, pp.400–407, 1995; H. Tamaki and H. Kita, "III — application to optimization 2 (in Japanese)," ditto, vol.39, pp.431–438, 1995; T. Iwamoto, "IV — theoretical analysis (in Japanese)," ditto, vol.39, pp.507–514, 1995; H. Tamaki, H. Kita and T. Iwamoto, "V — extensions (in Japanese)," ditto, vol.40, pp.69–76, 1996; H. Tamaki, H. Kita and T. Iwamoto, "VI — recent trends in evolutionary computation (in Japanese)," ditto, vol.40, pp.170–177, 1996.

[150] E. Tsang and C. Voudouris, "Fast local search and guided local search and their application to British Telecom's workforce scheduling problem," *Operations Re-*

search Letters, vol.20, pp.119-127, 1997.

[151] N.L.J. Ulder, E.H.L. Aarts, H.-J. Bandelt, P.J.M. van Laarhoven and E. Pesch, "Genetic local search algorithms for the traveling salesman problem," *Proc. 1st International Workshop on Parallel Problem Solving from Nature*, pp.109–116, 1990.

[152] S. Voß, S. Martello, I.H. Osman and C. Roucairol, eds., *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, 1999.

[153] C. Voudouris and E. Tsang, "Guided local search," Technical Report CSM-247, Department of Computer Science, University of Essex, 1995.

[154] C. Voudouris and E. Tsang, "Guided local search and its application to the traveling salesman problem," *European J. Operational Research*, vol.113, pp.469–499, 1999.

[155] J.P. Walser, *Integer Optimization by Local Search: A Domain-Independent Approach*, vol.1637 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 1999.

[156] M. Yagiura and T. Ibaraki, "Genetic and local search algorithms as robust and simple optimization tools," in: I.H. Osman and J.P. Kelly, eds., *Meta-Heuristics: Theory and Applications*, Kluwer Academic Publishers, pp.63–82, 1996.

[157] M. Yagiura and T. Ibaraki, "Metaheuristics as robust and simple optimization tools," *Proc. IEEE International Conference on Evolutionary Computation*, pp.541–546, 1996.

[158] M. Yagiura and T. Ibaraki, "The use of dynamic programming in genetic algorithms for permutation problems," *European J. Operational Research*, vol.92, pp.387–401, 1996.

[159] M. Yagiura and T. Ibaraki, "Metaheuristics as robust and simple optimization tools (in Japanese)," *Proc. Eighth RAMP Symposium*, pp.109–124, 1996.

[160] M. Yagiura and T. Ibaraki, "Analyses on the 2 and 3-flip neighborhoods for the MAX SAT," *J. Combinatorial Optimization*, vol.3, pp.95–114, 1999.

[161] M. Yagiura and T. Ibaraki, "Efficient 2 and 3-flip neighborhood search algorithms for the MAX SAT: experimental evaluation," *J. Heuristics* (to appear).

[162] M. Yagiura and T. Ibaraki, "Local search," a chapter in: P.M. Pardalos and M.G.C. Resende, eds., *Handbook of Applied Optimization*, Oxford University Press (to appear).

[163] M. Yagiura, T. Ibaraki and F. Glover, "An ejection chain approach for the generalized assignment problem," Technical Report #99013, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, 1999.

[164] M. Yagiura, T. Yamaguchi and T. Ibaraki, "A variable depth search algorithm for the generalized assignment problem," in: S. Voß, S. Martello, I.H. Osman and C. Roucairol, eds., *Meta-Heuristics: Advances and Trends in Local Search Paradigms*

for Optimization, Kluwer Academic Publishers, pp.459–471, 1999.

[165] M. Yagiura, T. Yamaguchi and T. Ibaraki, "A variable depth search algorithm with branching search for the generalized assignment problem," *Optimization Methods and Software*, vol.10, pp.419–441, 1998.

[166] T. Yamada and C.R. Reeves, "Landscape analysis of the flowshop scheduling problem and genetic local search (in Japanese)," *Transactions of Information Processing Society of Japan*, vol.39, pp.2112–2123, 1998.

[167] M. Yannakakis, "On the approximation of maximum satisfiability," *J. Algorithms*, vol.17, pp.475–502, 1994.

[168] M. Yannakakis, "Computational complexity," in: E.H.L. Aarts and J.K. Lenstra, eds., *Local Search in Combinatorial Optimization*, John Wiley & Sons, pp.19–55, 1997.

Mutsunori YAGIURA received B.E., M.E. and Ph.D. degrees in Engineering from Kyoto University, in 1991, 1993 and 1999, respectively. Since 1994, he has been with Department of Applied Mathematics and Physics, Kyoto University. Currently, he is a lecturer of Kyoto University. His interest includes combinatorial optimization, metaheuristics, discrete algorithms, computational complexity and their applications.

Toshihide IBARAKI received the B.E., M.E. and Ph.D. degrees in Engineering from Kyoto University, in 1963, 1965 and 1970, respectively. Since 1969, he has been with Department of Applied Mathematics and Physics, Kyoto University, except for two and a half years from 1983 to 1985, during which time he was with Department of Computer and Information Sciences, Toyohashi University of Technology. Currently, he is a professor of Kyoto University. He has held a number of visiting appointments with University of Illinois, University of Waterloo, Simon Fraser University, Rutgers University, University of Canterbury and others. He is the author of "Enumerative Approaches to Combinatorial Optimization," Baltzer AG., a coauthor of "Resource Allocation Problems: Algorithmic Approaches," MIT Press, and the author of several books in Japanese. His interest includes algorithms, optimization, computational complexity and their applications.