COMBINATORIAL OPTIMIZATION FOR SPACECRAFT SCHEDULING

William T. Scherer and Frank Rotman

Department of Systems Engineering
University of Virginia
Charlottesville, VA 22903

ABSTRACT

Most spacecraft have extremely limited resources compared with the state of the art in computer science and most missions have ambitious scientific goals, such as in the case of fly-bys like Voyager and Ulysses where there are limited windows of opportunity for achieving these goals. As a result, the scheduling of spacecraft experiments is a complex NP-complete problem for which an efficient solution procedure producing acceptable results is yet to be found. We describe the use of combinatorial optimization techniques applied to the automatic spacecraft scheduling problem. The sequencing problem is the search over the candidate sequences of experiments for a sequence that maximizes the value of the science conducted while minimizing constraint conflicts. Our exploratory computational results indicate that pseudo-random search techniques, such as simulated annealing, generate viable sequences in reasonable times.

1: Introduction

Mission operations for unmanned space research consist mainly of two parts called uplink and downlink. The downlink process involves collecting information from the spacecraft, decoding it, and sending it to the appropriate scientists and engineers. The uplink process is depicted in Figure 1 below. The scheduler receives instrument usage requests from scientists and spacecraft maintenance activity requests from engineers. These are integrated into a schedule of activities for the spacecraft which does not oversubscribe any spacecraft resource.

![Figure 1: The Uplink Process](image)

Creating a viable schedule is complex because most spacecraft have extremely limited resources compared with the state of the art in computer science, since they are designed years before they are launched, and only that hardware which has been designated as flight-qualified may be used in the designs. Also, there are strict size and weight constraints on the hardware. So limits on resources such as computer memory, processing time, power, and tape recorder capacity cause tight constraints on the scheduling of activities. In addition to limited spacecraft resources, most projects have ambitious scientific goals, and in the case of fly-bys like Voyager and Ulysses, there are limited windows of opportunity for achieving these goals. As a result, the spacecraft scheduling problem is a complex NP-complete problem for which an efficient solution procedure that produces acceptable results is yet to be found.

2: Problem statement

The spacecraft sequencing problem has as its primary goal the determination of a schedule of activities for a particular spacecraft. This problem can be modelled as a complex, large-scale combinatorial optimization problem [13]. Inherent complexities in the problem of determining a sequence include limited resources, time windows under which particular activities must occur, and complex interdependencies of various experiments and spacecraft resources. The large-scale nature of the problem is due to the number of potential activities (up to 1000 activities on a particular spacecraft load) and the number of spacecraft resources (on the order of 50 to 100). The number of candidate sequences increases exponentially with the number of activities that need to be scheduled [7]. However, many of these candidate schedules may not be feasible in the sense that they violate resource constraints, time window constraints, flight-rule constraints, or other constraints. The sequencing problem is the search over the candidate sequences for a sequence that maximizes the value of the science conducted while minimizing constraint conflicts.

Automated spacecraft scheduling is directly related to current planning systems which have automated conflict detection and sophisticated user interfaces. Plan Integrated Timelines (PLAN-IT) [2] is a scheduling tool used by JPL that has been used on various projects. The major success of the program has been its ability to enhance the
human schedulers' ability not only to produce an acceptable schedule more quickly than before but also to make adjustments to the schedule dynamically. The cognitive approach of encoding user-visualizations of constraints and a timeline view of the schedule makes this possible. PLAN-IT-II is an enhanced version recently completed by JPL researchers [6]. With the PLAN-IT-II capability now in place, it has become practical to begin exploring more advanced optimization techniques. The richness and completeness of PLAN-IT-II's representation and modeling capability allow a relatively easy adaptation to the new optimization techniques. The goal of our research is to develop the theoretical basis for enhancing such systems with automated conflict resolution which will eventually take the form of a user-invoked module that will be added to existing software. The completed system that we envision will save time for the human schedulers by performing the more mundane tasks in conflict resolution in considerably less time than is currently required.

A voluminous literature exists on planning and scheduling problems. Much of the research has been based on problems from the field of manufacturing systems [14]. While numerous advances have been made in this area, many researchers are looking towards less traditional approaches to problems of this nature in order to solve the large-scale problems often encountered in practical applications. Below we discuss the formulation of the spacecraft scheduling problem as a mathematical decision model and comment on various solution approaches.

3: Sequencing modeling background

The key elements of any mathematical model are the model parameters, the decision variables, the constraints, and the objective function [14]. For the spacecraft scheduling problem the parameters include the duration of each experiment, the resources required for each experiment, the resources available on the spacecraft, etc. The decision variables are the times at which various activities are to begin and end, along with the resource assignments of the activities. The constraints include time windows under which certain activities must occur, the amount of resources available at any given time, and various dependencies among activities. The objective is to maximize the value of the science experiments conducted while minimizing violations of the constraints. We note that the objective of automated scheduling systems is to minimize the amount of resource conflict rather than to completely eliminate conflict. Complete elimination of schedule conflicts is a highly political and volatile task which requires human intervention. Complicating issues in the modeling process include dynamic resource availability, dynamic resource requirements that are a function of the time the experiment begins, and additional restrictions on the scheduling of certain events.

A general model of the spacecraft sequencing problem can be formulated as a large, complex, linear integer programming model [13,3]. Such a model is robust in that an exact optimal solution to the problem can be obtained; however, the execution times become excessive as the number of activities increase. Current technology, in terms of computational power and algorithmic development, allows small problems with 20 or fewer activities to be solved in minutes. Problems with up to 100 activities could possibly be solved in hours. Large problems, with 500 or more activities, are intractable since the computational requirements increase exponentially with the number of activities [7,10]. This computational limitation has led researchers to investigate algorithms that search for good solutions since the time requirements are excessive for finding the best solution. These algorithms are often termed heuristics and are quite diverse in their approaches to searching. We believe that a viable solution approach will be a combination of various techniques for solving large-scale combinatorial optimization problems.

While the focus of this research is on a mathematical formulation of the spacecraft sequencing problem, we note that numerous researchers are investigating knowledge-based (KB) approaches to the sequencing problem [3,4,5,6,11]. These approaches often offer the advantages of KB systems, including natural language interfaces, expert compatible knowledge representations, and explanation capabilities. However, they are hindered by the inability to generate quality solutions in a reasonable time when used to solve large-scale combinatorial optimization problems. We envision that a final system for the aiding of sequencing would be an intelligent system that is hierarchical in nature, with the higher user interface levels being based on KB concepts employing optimization algorithms transparent to the system user. The focus of our effort is on exploring the viability of the proposed algorithmic approaches for sequencing automation.

4: Combinatorial optimization techniques

Standard techniques for solving large-scale combinatorial optimization problems include linear integer programming, heuristic search methods, and pseudo-random search methods. There are several novel solution approaches that we believe can be successfully integrated with mathematical programming techniques to create a new solution paradigm addressing large-scale spacecraft scheduling optimization problems. These are simulated annealing (SA) [1], random search, tabu search [8,9], and strategic oscillations. The heuristic search techniques operate in a straightforward fashion, as illustrated in Figure 2. The techniques differ in the way new sequences are obtained from old ones.

121
Figure 2: Search Flowchart

The power of these techniques lies in their ability to treat the objective function as a black box that returns a measure(s) of the goodness of the sequence; that is, these algorithms do not require a closed-form analytical description of the objective function nor any function derivatives. Goodness, or quality, of a solution is based on the value of the science in the sequence and a measure of the resource requirements and resource violations. We remark that acceptable measures of the goodness of a sequence are often difficult to determine.

Presently, four heuristics as well as combinations of these heuristics are being tested on the exploratory problem. All heuristics use the same procedure to find alternate schedules for consideration, but each heuristic uses its own criteria for acceptance and rejection of these new schedules. Basic descriptions of each heuristic follow.

The random hill climb heuristic is the simplest of the heuristics under consideration. Each time a new schedule is generated, the random hill climb procedure checks its objective function value, and if greater than that of the present schedule, the new schedule will replace the old one. Advantages to using this heuristic include speed and ease of implementation. Disadvantages include the possibility of being trapped at a local maximum and the heuristic’s lack of diverse solutions.

Instead of being greedy and going only in the direction of "better" solutions, the simulated annealing heuristic probabilistically accepts non-improving local solutions with the hope of finding a better global solution. The probability function that governs the SA's acceptance rate is the Boltzmann probability distribution which is based on two parameters - how much worse the candidate solution is from the present solution, and the present temperature. The first parameter, determined by the difference between objective function values, is a measure of how far the candidate solution has strayed (in value) from the present solution. As this parameter grows, the acceptance rate diminishes. The other parameter, temperature, decreases as time increases, and with its decrease comes a decreased probability of accepting a bad solution. The basic premise trying to be modeled is that you may have to give up a little value to eventually make future gains. Advantages to using the SA heuristic are its ability to overcome local optimal solutions and its ability to produce diverse solutions. Disadvantages include added evaluation time and added variable memory requirements.

Unlike the Random Hill Climb heuristic or the SA heuristic, Tabu Search is not a heuristic by itself; instead, it is a metaheuristic used to lead other heuristics to good candidate solutions [for example, see 13]. Although there are many forms of Tabu Search, the common link between all Tabu Searches is a tabu list containing information that is tabu, or not allowed. The list might be a series of moves recently made or values that were recently attained. Each time a new schedule is accepted, the tabu list is updated, with the newest information replacing the oldest information on the list. Tabu lists help the heuristic search in various ways, but common advantages gained by the addition of the tabu list include its ability to add constraints without changing present computer code, its ability to prevent cycling, and its ability to break out of local regions. Disadvantages to using the Tabu Search metaheuristic include its need for added variable memory as well as its drawback of adding evaluation time to the objective function.

Like Tabu Search, Strategic Oscillation is a metaheuristic that is used to lead other heuristics to good candidate solutions. Once a heuristic finds a local optimal solution, Strategic Oscillation forces the heuristic to accept a certain number of non-improving moves, regardless of how bad they are. This pushes the heuristic into a new local region, and it can then once again look for the local optimal solution of that region. The main disadvantage to using Strategic Oscillation is that each new region that is entered is not guaranteed to be better than the present region being examined; unfortunately, much time can be wasted by searching these fruitless regions.

5: Spacecraft sequencing modeling background

Below we describe an integer model for the spacecraft sequencing problem. Due to the complexity of the model, we generated a simplified linear model that allows for the testing of the heuristics. In order to present the mathematical model details we first describe the preliminary simplifying assumptions (PSAs) that reduce the computational complexity of the model. These features cannot be removed permanently from consideration for the sequencing problem being considered. They are, however,
being used in an approximation model for evaluating algorithms for the large-scale sequencing problem.

PSA1: Time is measured in discrete intervals.
PSA2: There is no synergy between experiments, i.e., the value of the experiments is considered as independent.
PSA3: Consideration is not made for delays in message sending or receiving (light time).
PSA4: Set-up costs are assumed to be built into the resource requirements.
PSA5: Repeated experiments are modelled as either independent experiments or captured in the definition of the resource requirement.
PSA6: Removal of the time dependency for the resource constraints.
PSA7: Limiting of the number of periods and use larger units of time, such as seconds.
PSA8: Limiting the number of experiments allowed.
PSA9: Depletable and renewable resources are eliminated from consideration.
PSA10: Assume zero time for any experiment set-up or build the set-up time into resource requirement.
PSA11: Assume that resources required are constant over the experiment.
PSA12: Assume resources required are independent of the time at which the experiment is activated.
PSA13: Assume that the duration of an experiment is independent of the start time.
PSA14: Assume that the resources available are constant over time.

Our preliminary model incorporates the above PSAs and is described below. This simplified model was used to replicate several simple sequencing problems to test the tractability, quality, and robustness of the heuristics described in this paper. The approximating model parameters are:

N: Number of experiments (index i used).
M: Number of resources (j).
T: Total number of discrete time units being scheduled (t).
R(i,j): Resources of type j required for experiment i.
RT(j): Resources available of type j throughout T.
V(i): Value of experiment i (utils; a term describing the unit value of where a 1 is maximum value and 0 is minimal value, V ∈ (0,1)).
ts(i): Lower bound on the time window for experiment i.
tf(i): Upper bound on the time window for experiment i.
d(i): Duration of experiment i.
wu(j): The "weight" associated with under utilizing resource j (normalized to one).
wO(j): The "weight" associated with over utilizing resource j (normalized to one).

c(o): The "weight" associated with the components of the objective function (o=1,3).
CP: The penalty term for violation of a "hard" constraint.

The approximating model decision variables are:

x(i): Flag indicating activation of experiment i (0 = non-activated, 1 = activated).
b(i): Time at which the replication n for experiment i is activated.

Our objective function is a surrogate measure of the primary goal "maximize the amount of science," where we maximize the measure of fitness of a sequence (i.e., quality of a schedule), where fitness is described by the weighted sum of three objectives plus a penalty term for violation of constraints:

(i). The value of the included experiments, \( V = \sum_i x(i) \cdot V(i) \)

(ii). The measure of resource utilization, \( U = \sum_j \left( \sum_t \left( RU(j,t) / RT(j) \right) \cdot wu(j) \right) / M \) where \( RU(j,t) \) is the amount of resource j used at time \( t \) by the given schedule, for \( RU < RT \) (i.e., a measure of the utilization for a given schedule)

(iii). The measure of sequence feasibility (the inverse of sequence conflict), \( O = \sum_j \left( \sum_t \left( RT(j) / RU(j,t) \right) \right) / T^* \cdot wO(j) \) \( / M \) for \( RT < RU \)

(iv). The penalty for violation of constraints, \( P = \sum_k \left( \left(\text{constraints} \right)^* (CP) \right) \) (summation over k violated constraints)

Note that RU is calculated using the schedule, durations, and resource requirements:

\( RU(j,t) = \sum_i \ x(i) \cdot R(i,j) \cdot \left[ 1 - b(i) \right] \geq 0 \text{ and } < d(i) \).

In an expanded model, the calculation of \( RU \) and \( RT \) can get quite complex and possibly nonlinear. Also, \( V, U, \) and \( O \) are scaled in the above expressions to be between 0 and 1, and all are of the nature "more is better." The complete objective function therefore is:

maximize \( G = c(1) \cdot V + c(2) \cdot U + c(3) \cdot O - P \)

where the c's are weights on the three objectives and the \( \sum c(o) = 1 \). We note that this a rather arbitrary objective function; however, the weights (c's and w's) allow adjustment of the relative importance of issues such as resource conflict versus science per unit time. The only constraints (albeit critical) in the approximating model are the time window constraints.

123
6: Implementation

To test the heuristics, three similar problem types were developed. All problem types have the following characteristics:

- One day scheduled in 5 minute intervals (288 time units)
- 300 experiments are to be scheduled
- 3 resources were available for each experiment
- Each resource had 5 to 10 units available at each time unit
- Experiments had a 50% chance of needing each resource
- Each experiment required up to 30% of the maximum available resource units for those resources that it needed
- 20% of the experiments could be scheduled at anytime as long as these experiments were be completed on or before time unit 288
- The remaining experiments had a random time window
- The duration of each experiment ranged from 30 minutes to 2 hours (6 to 24 time units)
- Each experiment had an associated experiment value between 1 and 10
- The objective function weights (ci's) were 2/7, 1/7, and 4/7 respectively.

The problem specifics are as follows:

Problem Type #1 - A very large penalty was subtracted from the objective function if, at any time, any of the resources were overutilized. The penalty was proportional to the amount the resources were overutilized, but still was very large even for minimal overutilization. Although some of the heuristics allowed this heavily penalized schedule to be accepted with a very small (almost negligible) probability, it should be noted that all generated schedules were feasible.

Problem Type #2 - A very large penalty was subtracted from the objective function if, at any time, any of the resources were overutilized by more than 12%. The normal objective function penalty remained for all resource overutilization up to 12%. The schedules resulting from this constraint relaxation were generally feasible, but had slightly overprescribed resources at a few blocks of time. This problem type closely mimics the real-world scheduling problem, where small conflicts are to be eliminated later through negotiations.

Problem Type #3 - The normal objective function penalty was in effect for all resource overutilization. This complete relaxation of the penalty function caused schedules to be created that had slightly overutilized resources at some blocks of time, and had moderately overutilized resources in a few time regions.

7: Computational results

Both the simplified model and the heuristics previously discussed were coded in ANSI C in the interest of portability and flexibility. For the testing of the heuristics, only the default parameters were used. For each of the 3 problem types, 10 problems were generated. Then, the heuristics were individually run for 15 minutes of CPU time (on an IBM RISC 6000) for each of the 10 problems. Heuristic time profiles, objective function values, experiment values, and utilization statistics were collected for all runs. Based on the data tables, the following conclusions can be drawn:

Problem Type #1. The heuristics implementing the tabu meta-heuristic achieved the highest objective function values for 8 out of the 10 problems. The pure simulated annealing heuristic found the highest objective function values for the other 2 problems. For all 10 problems, adding the tabu meta-heuristic to the random hill climb heuristic caused an increase in the objective function values from that of the random hill climb heuristic alone. Therefore, for these problems, the random hill climb with tabu heuristic can be said to dominate the pure random hill climb heuristic. For 7 of the 10 problems, adding the tabu meta-heuristic to the simulated annealing heuristic caused an increase in the objective function values from that of the pure simulated annealing heuristic. Additionally, it was found that, on average, the random hill climb with tabu heuristic came closest to achieving the highest objective function values found by the five tested heuristics. These averages ranged from 98.13% (random hill climb) to 99.68% (random hill climb with tabu). The tables at the end of the paper illustrate these results, where the objective function values have been normalized by the maximum value achieved over all algorithms.

On this highly constrained type of spacecraft scheduling problem, the tabu meta-heuristic works quite efficiently. Examining the example time profile (see Figure 3) shows that the implementation of the tabu meta-heuristic causes a noticeable increase in the slope of the objective function value versus CPU time curve.

Problem Type #2. The heuristics implementing the tabu meta-heuristic achieved the highest objective function values for 6 of the 10 problems. For 7 of the 10 problems, the strategic oscillation meta-heuristic outperformed the random hill climb heuristic. With problem type #1, the random hill climb with tabu heuristic dominated the pure random hill climb heuristic; however, the addition of the tabu meta-heuristic to the random hill climb heuristic only caused an increase in the objective function values for 6 of the 10 problems. Adding the tabu meta-heuristic to the simulated annealing heuristic only caused an increase in the objective function values for 4 of the 10 problems. After normalizing the
The tabu meta-heuristic was designed to work efficiently on highly constrained problems, and because this type of problem is somewhat relaxed from the pure problem (fully feasible), the tabu meta-heuristic does not perform as efficiently as it did with the problems of problem type #1.

Problem Type #3. The heuristics implementing the tabu meta-heuristic did not find the highest objective function values for any of the problems. This is further proof that the tabu meta-heuristic tends to work better on highly constrained problems than it does on very relaxed problems. The strategic oscillation meta-heuristic outperformed the random hill climb heuristic for 9 of the 10 problems (see Table 3). After normalizing the objective function values (as described in the results of problem type #1) it was found that, on average, the strategic oscillation meta-heuristic came closest to achieving the highest objective function values found by the five tested heuristics. These averages ranged from 99.03% (random hill climb with tabu) to 99.78% (strategic oscillation).

As was stated earlier, the tabu meta-heuristic was designed to work efficiently on highly constrained problems, and because this type of problem is extremely relaxed from the pure problem, the tabu meta-heuristic does not perform well at all. In fact, the heuristics employing the tabu meta-heuristic had the lowest normalized averages with respect to objective function values. Unlike the tabu meta-heuristic, the less constrained the problem, the greater the chance that the strategic oscillation meta-heuristic will randomly proceed to better regions. As was expected, the strategic oscillation meta-heuristic's ability to search diverse regions works quite well given this loosely constrained type of scheduling problem.

8: Conclusions

Specific observations and conclusions are as follows:

- Fifteen minutes of CPU time is not necessarily required in order to generate "good" sequences. Stopping criteria, such as those based on the gradient of the objective function versus time plot, should be used in order to indicate when the (decreasing) marginal returns indicate that further iterations are not worth the CPU time required.
- The time results are dependent on the machine being used, and changes such as a new chip, can have significant impacts on the absolute time requirements. We do believe, however, that the relative performances would not significantly change unless there was a major architecture change, such as the implementation on a parallel computer.
- The preferred algorithmic approach is likely to be a combination of a heuristic initial schedule generator, possibly expert system based, followed by a mixture of pseudo-random search techniques. Generation of a high-quality initial schedule is likely to improve the time performance of the heuristics. Techniques for generating initial schedules are a topic of future research.
- There is the possibility that for different problem data structures, different heuristics may be preferred. One possibility would be to develop an expert system that identifies type of problem and guides the user on the selection of what heuristics to employ.
- Actual spacecraft sequencing data will be used in future tests. A better understanding of actual data and more in-depth knowledge of how the scheduling process works will allow the capture some of the experts' knowledge.

Numerous combinatorial optimization approaches have been examined for the spacecraft sequencing problem. We remark that this evaluation is strictly exploratory, i.e., the goal is to determine the applicability of certain approaches to the spacecraft scheduling problem. Our computational results indicate that pseudo-random search techniques, such as simulated annealing, generate viable sequences in reasonable times. Although the spacecraft scheduling problems used for testing were not based on actual experiment request data, they did match the data structures of actual problems with regard to size and constraint types. Therefore, we believe that the techniques described in this paper represent viable approaches to spacecraft sequencing problems.

Acknowledgments: The authors would like to thank the considerable assistance of Sven Grenander, Curt Eggemeyer, and Dr. Carol Collins of the Sequence Automation Research Group in the Mission Profile and Sequencing Section at the Jet Propulsion Laboratory in Pasadena, California.
Table 1: Problem Type #1 - Normalized Objective Function Values

<table>
<thead>
<tr>
<th>Problem Number</th>
<th>Random Hill Climb</th>
<th>Simulated Annealing</th>
<th>Random Hill Climb W/Tabu</th>
<th>Simulated Annealing W/Tabu</th>
<th>Strategic Oscillation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9781</td>
<td>0.9876</td>
<td>1.0000</td>
<td>0.9994</td>
<td>0.9844</td>
</tr>
<tr>
<td>2</td>
<td>0.9785</td>
<td>0.9880</td>
<td>1.0000</td>
<td>0.9949</td>
<td>0.9743</td>
</tr>
<tr>
<td>3</td>
<td>0.9718</td>
<td>0.9900</td>
<td>0.9848</td>
<td>1.0000</td>
<td>0.9848</td>
</tr>
<tr>
<td>4</td>
<td>0.9901</td>
<td>0.9982</td>
<td>1.0000</td>
<td>0.9977</td>
<td>0.9885</td>
</tr>
<tr>
<td>5</td>
<td>0.9787</td>
<td>0.9869</td>
<td>0.9946</td>
<td>1.0000</td>
<td>0.9828</td>
</tr>
<tr>
<td>6</td>
<td>0.9896</td>
<td>1.0000</td>
<td>0.9977</td>
<td>0.9855</td>
<td>0.9888</td>
</tr>
<tr>
<td>7</td>
<td>0.9731</td>
<td>0.9858</td>
<td>1.0000</td>
<td>0.9887</td>
<td>0.9731</td>
</tr>
<tr>
<td>8</td>
<td>0.9868</td>
<td>0.9888</td>
<td>1.0000</td>
<td>0.9893</td>
<td>0.9926</td>
</tr>
<tr>
<td>9</td>
<td>0.9908</td>
<td>0.9967</td>
<td>0.9979</td>
<td>1.0000</td>
<td>0.9912</td>
</tr>
<tr>
<td>10</td>
<td>0.9752</td>
<td>1.0000</td>
<td>0.9981</td>
<td>0.9933</td>
<td>0.9726</td>
</tr>
</tbody>
</table>

Average Percent Of Best: 98.13% 99.22% 99.68% 99.50% 98.31%

Table 2: Problem Type #2 - Normalized Objective Function Values

<table>
<thead>
<tr>
<th>Problem Number</th>
<th>Random Hill Climb</th>
<th>Simulated Annealing</th>
<th>Random Hill Climb W/Tabu</th>
<th>Simulated Annealing W/Tabu</th>
<th>Strategic Oscillation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9903</td>
<td>0.9900</td>
<td>0.9900</td>
<td>0.9896</td>
<td>1.0000</td>
</tr>
<tr>
<td>2</td>
<td>0.9966</td>
<td>0.9980</td>
<td>1.0000</td>
<td>0.9890</td>
<td>0.9939</td>
</tr>
<tr>
<td>3</td>
<td>0.9969</td>
<td>1.0000</td>
<td>0.9934</td>
<td>0.9931</td>
<td>0.9853</td>
</tr>
<tr>
<td>4</td>
<td>0.9953</td>
<td>1.0000</td>
<td>0.9973</td>
<td>0.9952</td>
<td>0.9948</td>
</tr>
<tr>
<td>5</td>
<td>0.9830</td>
<td>0.9990</td>
<td>0.9830</td>
<td>1.0000</td>
<td>0.9891</td>
</tr>
<tr>
<td>6</td>
<td>0.9916</td>
<td>0.9905</td>
<td>1.0000</td>
<td>0.9905</td>
<td>0.9992</td>
</tr>
<tr>
<td>7</td>
<td>0.9919</td>
<td>0.9962</td>
<td>1.0000</td>
<td>0.9942</td>
<td>0.9996</td>
</tr>
<tr>
<td>8</td>
<td>0.9933</td>
<td>0.9963</td>
<td>0.9934</td>
<td>0.9924</td>
<td>1.0000</td>
</tr>
<tr>
<td>9</td>
<td>0.9823</td>
<td>0.9932</td>
<td>0.9927</td>
<td>1.0000</td>
<td>0.9916</td>
</tr>
<tr>
<td>10</td>
<td>0.9770</td>
<td>0.9882</td>
<td>0.9835</td>
<td>1.0000</td>
<td>0.9793</td>
</tr>
</tbody>
</table>

Average Percent Of Best: 99.00% 99.51% 99.33% 99.44% 99.33%

Table 3: Problem Type #3 - Normalized Objective Function Values

<table>
<thead>
<tr>
<th>Problem Number</th>
<th>Random Hill Climb</th>
<th>Simulated Annealing</th>
<th>Random Hill Climb W/Tabu</th>
<th>Simulated Annealing W/Tabu</th>
<th>Strategic Oscillation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9917</td>
<td>1.0000</td>
<td>0.9826</td>
<td>0.9854</td>
<td>0.9962</td>
</tr>
<tr>
<td>2</td>
<td>0.9910</td>
<td>0.9908</td>
<td>0.9900</td>
<td>0.9932</td>
<td>1.0000</td>
</tr>
<tr>
<td>3</td>
<td>1.0000</td>
<td>0.9931</td>
<td>0.9986</td>
<td>0.9978</td>
<td>0.9999</td>
</tr>
<tr>
<td>4</td>
<td>0.9954</td>
<td>1.0000</td>
<td>0.9977</td>
<td>0.9970</td>
<td>0.9975</td>
</tr>
<tr>
<td>5</td>
<td>0.9944</td>
<td>1.0000</td>
<td>0.9943</td>
<td>0.9948</td>
<td>0.9998</td>
</tr>
<tr>
<td>6</td>
<td>0.9908</td>
<td>0.9987</td>
<td>0.9859</td>
<td>0.9885</td>
<td>1.0000</td>
</tr>
<tr>
<td>7</td>
<td>0.9875</td>
<td>0.9963</td>
<td>0.9984</td>
<td>0.9916</td>
<td>1.0000</td>
</tr>
<tr>
<td>8</td>
<td>0.9881</td>
<td>1.0000</td>
<td>0.9910</td>
<td>0.9942</td>
<td>0.9910</td>
</tr>
<tr>
<td>9</td>
<td>0.9907</td>
<td>0.9913</td>
<td>0.9798</td>
<td>0.9862</td>
<td>1.0000</td>
</tr>
<tr>
<td>10</td>
<td>0.9901</td>
<td>1.0000</td>
<td>0.9802</td>
<td>0.9828</td>
<td>0.9933</td>
</tr>
</tbody>
</table>

Average Percent Of Best: 99.20% 99.70% 99.03% 99.12% 99.78%
**References**


