Mat-2.108 Independent research projects in applied mathematics

Expert selection problem – evaluation of cost efficiency approach

25 March 2002

Helsinki University of Technology
Department of Engineering Physics and Mathematics
Systems Analysis Laboratory
Jarno Pelkonen
44441A
Abstract

This paper presents a way to formulate expert selection problem as a nonlinear 0-1 programming problem. In expert selection problem one needs to select a balanced set of experts with limited budget from large pool of experts. Approximate Cost-efficiency selection (CES)-heuristic is presented and its capability to optimize the formulated nonlinear 0-1 programming problem is studied.

CES-heuristic was tested by comparing it to other exact algorithms and approximate heuristics. The comparisons were done with generated test data that simulates real data. CES-heuristic was tested against large number of different sized problems. The size of expert pool in question varied from ten to many thousands. Problems and costs related to gathering real data were the reasons to use generated test data.

CES-heuristic was found to be accurate enough (better than 95% of the optimum) with reasonable sized problems (panel size larger than 5). Also its performance in terms of computing time showed to be excellent, even in the largest problems (2 seconds in average with problems size of 500 experts with normal PC hardware).

Anyhow, it gave worse results than other four heuristics tested. The differences were also statistically significant. CES-heuristic’s propensity to stop after first local optima found makes it unsuitable for solving general nonlinear 0-1 programming problems. CES-heuristic is also regarded as a weak heuristic research literature [11].

The real-world applicability of CES-heuristic remained open. CES-heuristic proved to be a good heuristic to solve the formulated mathematical problem, but whether the problem it is solving is relevant seemed questionable. According to the author’s judgment the CES-heuristic does not help in real-world expert selection problem.
Table of contents

ABSTRACT 2

TABLE OF CONTENTS 3

1 INTRODUCTION 5
  1.1 DOCUMENT INTRODUCTION 5
  1.2 EXPERT SELECTION CONTEXT 5
  1.3 RESEARCH PROBLEM DEFINITION 5
  1.4 RESEARCH SCOPE 6
  1.5 RESEARCH METHODOLOGY 6
  1.6 DOCUMENT STRUCTURE 6

2 MATHEMATICAL FORMULATION 6
  2.1 MATHEMATICAL PROBLEM DEFINITION 6
  2.2 INTRODUCTION TO THE CES HEURISTIC 7
  2.3 REVIEW OF NONLINEAR 0-1 PROGRAMMING RESEARCH 8
  2.4 COMPARISON ALGORITHMS 11

3 ALGORITHM TESTING 15
  3.1 APPROACH 15
  3.2 TEST DATA 15

4 RESULTS 17
  4.1 ACCURACY 17
  4.2 SENSITIVITY ANALYSIS 24
  4.3 COMPUTING EFFICIENCY 27
  4.4 ANALYSIS OF THE RESULTS 29
5 REAL-WORLD APPLICABILITY AND FINAL CONCLUSIONS

REFERENCES

6 APPENDICES

6.1 RECURSIVE SIMULATED ANNEALING
1 Introduction

1.1 Document introduction

This paper presents a way to formulate expert panel selection problem as a nonlinear 0-1 programming problem and Cost-Efficiency Selection (CES)-heuristic can be used to optimize the formulated problem. CES-heuristic’s capability to optimize the formulated nonlinear 0-1 programming problem is studied.

The CES-heuristic is an approximate heuristic. Its accuracy is measured and compared to other known approximate heuristics and accurate algorithms based on generated test data. The heuristic's applicability to real-world problems is also discussed.

1.2 Expert selection context

In expert selection problem it is question about selecting balanced set of stakeholders according certain criteria. Expert selection problem rises when one must select members to an expert panel that will be formed to support decision-maker on complex problem field, for example. One concrete case is forming an expert panel for technology foresight study. To get multiple perspectives into the study, it is desirable to select experts with different backgrounds. One often needs experts with backgrounds of environmental and social sciences, economists and experts with business knowledge - in addition to experts having deep, but often narrow knowledge over particular technology area. In real world these kinds of panels are usually formed to support governmental decision-making process at national or international level.

Depending on expert panel the number of applicants varies. Interviews and ad-hoc procedures can be used if the number of applicants is low. But if the number of applicants is in tenths, hundreds or even in thousands, one may need tools for selecting the best set of experts.

1.3 Research problem definition

This paper has three objectives:

1. To investigate how good results does the CES-heuristic give. The accuracy is measured as a percentage of the global optimum/best known solution.

2. To investigate how parameters affect to the accuracy of the CES-heuristic.

3. To investigate the performance of the CES-heuristic in sense of computational efficiency.
1.4 Research scope

This paper focuses on judging CES-heuristic’s accuracy in sense of results provided. In addition to that a brief analysis over CES-heuristic’s real-world applicability is given.

1.5 Research methodology

Testing of the CES-heuristic was done by comparing its results to results given by other exact algorithms and approximate heuristics. The comparisons were done with generated test data that simulates real data. The heuristic was tested against large number of different sized problems. The size of expert pool in question varied from ten to many thousands. Problems and costs related to gathering real data were the reasons to use generated test data.

Because of the hard nature of the underlying mathematical problem [1], the CES-heuristic was tested against other approximate heuristics in larger problems. Author lacked access to special software and hardware capable of solving nonlinear 0-1 programming problems exactly. With smaller problems, brute-force-method was used to calculate optimum expert set, but with larger problems, the CES-heuristic was compared to other approximate heuristics. Heuristics’ accuracy was measured as percentage of the optimum/best result found.

1.6 Document structure

Chapter 2 presents comparison heuristics used and mathematical formulation for the expert selection problem. Chapter 3 describes how test data was generated and what were the assumptions used when generating the test data. Chapter 4 presents test results and statistical analysis of CES-heuristic’s performance. The CES-heuristic was tested for accuracy, parameter sensitivity and performance. Chapter 5 discusses CES-heuristic’s real-world applicability and draws final conclusions.

2 Mathematical formulation

This section presents reader mathematical formulation of the expert selection problem and the comparison heuristics used.

2.1 Mathematical problem definition

In this study, expert selection problem was modeled as a nonlinear 0-1 (binary integer) maximization problem, where each expert had fixed cost and each expert was evaluated
against common criteria. The objective was to maximize expert set’s utility within given budget constraints. This formulation has been presented by Salo[2].

The utility of selected expert set on certain criterion can be measured as value of their accumulated expertise (i.e. sum of their scores over the criterion). In order to model decreasing marginal utility (i.e. each criterion’s marginal utility decreases as more experts are selected), a concave utility function was used for each criterion. To calculate the total utility, each criterion was weighted and summed together.

Mathematical formulation of the expert’s selection problem is:

$$\max \sum_i w_i f_i \left( \sum_k v_{ik} x_k \right) \quad \text{so that,}$$

$$\sum_k c_k x_k \leq \text{budget} \quad (2)$$

Where:

- $$x_k$$ is decision variable ($$x_k$$ is 1 if k-th’s expert is selected, 0 if not)
- $$f_i$$ is a concave value function associated to i-th criterion
- $$c_k$$ is cost related to k-th expert
- $$v_{ik}$$ is k-th expert’s score on the i-th criterion.
- $$w_i$$ is i-th’s criterion’s weight

**Equation 1. Formulation of expert selection problem**

The optimization problem has linear constraints, ($$\sum_k c_k x_k \leq \text{budget}$$), but objective function $$f$$ is nonlinear. In this study, ln(1+x) was selected to present the concave utility function. This selection was considered reasonable and not limiting the heuristic’s real-world applicability. The complexity of the problem increases exponentially with the problem size.

2.2 Introduction to the CES heuristic

The CES-heuristic is a simple iterative heuristic. It selects experts iteratively based on their cost-efficiency, one at each round until no expert can be chosen in budget constraints. The heuristic has three steps [2].
• Step 1. At first, assume that no experts have been selected.

• Step 2. For each unselected expert, compare ratio between the 1) incremental improvement of the objective function (computed under assumption that the expert is added to the current set of experts), and 2) the cost of selecting this expert.

\[
\text{ratio} = \frac{\text{value}_{\text{cur} + 1} - \text{value}_{\text{cur}}}{\text{cost}_{\text{cur} + 1} - \text{cost}_{\text{cur}}}
\]

(3)

• Step 3. Choose the expert (within budget constraints) for whom this ratio is greatest. If an expert was found, add him or her to the current set of experts and go back to step 2, if not, terminate the heuristic and return the current set of experts.

Figure 1 below elucidates how CES–heuristic optimizes expert selection problem (red line) at cost-value coordination. The expert sets at the figure are taken randomly (yellow points). The CES-heuristic is known as *Steepest-Ascent-One-Point-Move* heuristics presented by Reiter and Rice in 1966 [REITER] in research literature.

Figure 1. Convergence of the Cost-efficiency selection heuristic

2.3 Review of nonlinear 0-1 programming research

Since 50's zero-one (0-1) programming has been utilized in operations research problems involving qualitative decision variables (Hansen [5]). The field of 0-1 programming is hard, particularly when either or both constraints or objective function is nonlinear. Hansen [5] categorizes constrained nonlinear 0-1 programming algorithms into four categories; linearization, enumeration, cutting-plane and stochastic algorithms. In addition to these algorithms, approximate heuristics have been developed to solve large problems.
Only few researchers formulate nonlinear 0-1 programming problems in other than *multilinear* form (5) Balas[6], Hansen[5]. Theorem presented by Hammer [7] states that every constrained nonlinear 0-1 programming problem can be expressed in multilinear form.

$$\max f(x) = \sum_{k=1}^{N} c_k T_k$$  \hspace{1cm} (5)$$

Subject to:

$$g_i(x) = \sum_{k=1}^{N} a_{ik} T_k \leq b_i \hspace{0.5cm} i = 1,2,...,m$$  \hspace{1cm} (6)$$

Where:

$$T_k = \prod_{j \in N_k} x_j, \hspace{0.5cm} N_k \subseteq N = \{1,2,...,n\}, \hspace{0.5cm} k = 1,2,...,p$$

and

$$T_{ik} = \prod_{j \in N_{ik}} x_j, \hspace{0.5cm} N_{ik} \subseteq N, \hspace{0.5cm} k = 1,2,...,p, \hspace{0.5cm} i = 1,2,...,m.\hspace{1cm}$$

In linearization approach products $T_k$ (or) $T_{ik}$ are replaced with new 0-1 variables $x_{n+k}$ and new constraints (7 & 8) are added. By this way the problem can be changed into linear form and solved as constrained linear 0-1 programming problem.

$$\sum_{j \in N_k} x_j - x_{n+k} \leq |N_k| - 1$$  \hspace{1cm} (7)$$

and

$$- \sum_{j \in N_k} x_j + |N_k| x_{n+k} \leq 0$$  \hspace{1cm} (8)$$

However, linearization approach has drawback in this particular case, since linearization of the expert selection problem formulated increases the number of 0-1 variables significantly (12).

Expert selection problem formulation:

$$\max f(x) = \sum_{j} c_j \ln(1 + \sum_{i} x_i v_{ij}) \iff$$

$$\max e^{f(x)} = e^{\sum_{j} c_j \ln(1 + \sum_{i} x_i v_{ij})} \iff$$

$$\max e^{f(x)} = \prod_{j} (1 + \sum_{i} x_i v_{ij})^{c_j}$$  \hspace{1cm} (11)$$
From (11) it follows that there will be a large number of products of 0-1 variables $x_i$. If the products are linearized, new 0-1 variables need to be introduced. The number of 0-1 variables in linearized form follows approximately (12).

Number of 0-1 variables in linearized expert selection problem

\[
\sum_{k=1}^{i} \frac{i!}{k!(i-k)!} = \sum_{k=1}^{i} \binom{i}{k}
\]  

(12)

Linearization and cutting-plane algorithms linearize objective function and constraints first. The increased number of 0-1 variables may hinder application of these algorithms into the formulated expert selection problem. Hansen [5] mentions that the problem of increased number of 0-1 variables hinders algorithms based on linearization in general.

Enumeration methods solve the problem with branch-and-bound algorithm. According to tests run by Hansen [5], enumeration methods appear to perform the best when compared to other algorithms.

Because of the hard nature of the underlying mathematical problem, heuristics have been developed to solve large nonlinear 0-1 programming problems. CES-heuristic presented in this paper is called steepest-ascent-one-point-move heuristic (presented by Reiter and Rice [9]) in research literature. More advanced variation of it is Steepest-ascent-mildest-descent (SAMD) heuristic of Jaumard, Hansen and Poggi de Aragão[10]. In this variation a move from local optima is allowed along the direction of mildest and reverse change is forbidden for a given number of iterations. The procedure terminates if no improvement in the value of best solution is observed for a given number of iterations [11]. Glover has proposed similar approach under name of Tabu search [12]. Various researchers have reported SAMD performing well in various nonlinear 0-1 programming problems [11].

Stochastic method called Simulated annealing of Kirpatrick, Gelatt and Vecchi [13] have proved to be competitive for some classes of 0-1 programming problems. In this approach, a local change is randomly obtained and its effect on the value of the objective function is evaluated. If there is an improvement the local change is accepted. If there is deterioration, a probability for acceptance is calculated. The probability for acceptance decreases as the amount of deterioration and the time elapsed (or local changes already considered)[11]. This heuristics simulates slow cooling of substance.
2.4 Comparison algorithms

CES-heuristic was compared with the following algorithms/heuristics.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th># of experts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brute-force</td>
<td>Testing all possible expert sets</td>
<td>&lt; 20</td>
</tr>
<tr>
<td>Monte-Carlo selection</td>
<td>Random selection of expert sets</td>
<td>&lt; 30</td>
</tr>
<tr>
<td>Stepwise elimination</td>
<td>Iterative steepest-ascent &amp; mildest-descent heuristics. Heuristic stops after cyclical pattern is detected.</td>
<td>&lt; 1500</td>
</tr>
<tr>
<td>Steepest-Ascent-Mildest-Descent</td>
<td>Heuristics that allows moves away from local optima and prevents return to the local optima for fixed number of iterations.</td>
<td></td>
</tr>
<tr>
<td>Simulated annealing</td>
<td>Stochastic heuristic with random and conditional one-point move from local optima.</td>
<td></td>
</tr>
<tr>
<td>Recursive Simulated annealing</td>
<td>Modified version of Simulated Annealing heuristic.</td>
<td></td>
</tr>
<tr>
<td>Continuous optimization + rounding</td>
<td>Optimizing equivalent, but continuous problem with Sequential Quadratic Programming – algorithm (and rounding the results).</td>
<td>&lt; 150</td>
</tr>
</tbody>
</table>

Table 1. Benchmark algorithms used

2.4.1 Brute-force

Finding global optimum for nonlinear integer problem is hard. Brute-force method – i.e. computing through all possible expert sets – can find the optimum value, but it does not scale due to the fact that the number of different alternatives (expert sets) increases in $2^N$ as the number of experts (N) increases. Therefore, brute-force method could solve expert selection problem in reasonable time only when the problem was small (N < 20).

2.4.2 Monte-Carlo method

For problems larger than brute-force method could handle, Monte-Carlo method was tried. The idea of Monte-Carlo method is to generate expert sets randomly from given probability distribution and select the best expert set.
In brute-force method, the computations are done for many expert sets that do not fit into budget constraints. By forming such a probability distribution that provides optimum sized expert sets, the computation effort can be focused on the most relevant expert sets. Therefore, Monte-Carlo method could handle slightly larger problems than brute-force method.

Monte-Carlo method was used to create expert sets with the following way:

1. Minimum and maximum values for feasible number of experts in expert set were calculated.

   \[
   \begin{align*}
   \text{min} &= \text{ceil}(\text{budget} / \text{max(Experts' cost)}) \\
   \text{max} &= \text{floor}(\text{budget} / \text{min(Experts' cost)})
   \end{align*}
   \]

2. Uniform distribution was used to select \( n \), number of experts in expert set, between \( \text{min} \) and \( \text{max} \).

3. Uniform distribution was used to select \( n \) number of experts from the expert pool.

This procedure allowed marking off many infeasible expert sets, but not all. See Figure 2 for result of marking off experts that certainly are infeasible. It must be noted that this method of generating random expert sets is not optimal.

Figure 2. Monte-Carlo method

The computing power available limited the maximum number of Monte-Carlo iterations to \(~2^{25}\). Test results showed that Monte-Carlo method was rather unusable for problems larger that 30 experts. Because the bad performance of the Monte-Carlo method it was not included among the group of comparison algorithms used when computing test statistics.
Monte-Carlo method was used for getting random initial values for stepwise selection/elimination heuristics.

2.4.3 Stepwise elimination/selection with random initial values

This heuristic functions as CES-heuristic when the cost of current expert set is in the budget. After the cost of expert set exceeds budget constraint, expert with the worst utility/cost ratio is removed from the expert set. This elimination continues until the cost of the expert set is in budget again. This iteration lasts until a circular pattern is detected. This heuristic was initiated with random expert set. Figure 3 shows how the Stepwise selection/elimination -heuristic converges from random point to its optimum.

![Figure 3. Convergence of Stepwise selection/elimination heuristic](image)

**Figure 3. Convergence of Stepwise selection/elimination heuristic**

2.4.4 Steepest-Ascent-Mildest-Descent heuristics (SAMD)

Steepest-Ascent-Mildest-Descent (SAMD) heuristics takes advantage of the direction of steepest-ascent in objective function as the CES-heuristic does. The difference is that it does not stop after finding the first local optima. Once it finds local optima it moves to direction along the mildest descent. The heuristic blocks moves back to the local optima for certain number of iterations to avoid cycling.

Two (2) different variations of the SAMD were tried in this study. Blocking the moves that reverse moves done along mildest descent was done as presented in [11]. After a move to descent direction a move backwards was blocked for fixed number of iterations. This prevents cycling in to some extent and allows the heuristic to get out of local optima. Glover presents different strategies for cycling avoidance in his Tabu search paper [12].

Researchers have reported SAMD to give good results in various applications. It has proved to be more computing efficient compared to stochastic heuristics.
Two versions of the SAMD were tested. One chose experts according absolute improvement in objective function’s value, another according to marginal value/cost ratio.

2.4.5 Simulated annealing

Simulated Annealing (SA) heuristics has its background in thermodynamics. It simulates a collection of atoms initially in equilibrium at high temperature and subjected to slow cooling. It is known that rapid cooling blocks the system in disordered high-energy state whereas slow cooling will bring the system into ordered slow energy state [11].

The SA-heuristic allows moves away from local optima with certain probability. The probability depends in the amount of decrease in value of the objective function and “system temperature”. When the system “cools down” (=heuristic runs forward) the probability of allowing deteriorative local move diminishes. Moves away from local optima are allowed with probability:

\[ P = \exp(\text{deterioration in value of the objective function/system temperature}) \] (5)

In this study the SA-heuristic was implemented so that the system temperature gets lower by fixed factor at each round. Simultaneously the probability to accept moves away from local optima diminishes. The heuristics stops after no moves have been accepted for certain amount of rounds.

In this study the SA heuristic was modified a bit to guarantee reasonable computing time. A temperature limit was defined and after system temperature got under this limit, the heuristic was terminated. This modification enabled setting an upper limit for the time the heuristic runs.

A good overview of the Simulated Annealing heuristics is available at


2.4.6 Recursive Simulated annealing

To overcome SA’s propensity to accept bad moves at early stages, a modified version of the SA heuristic was developed. The modification showed to perform well and run faster than standard SA because the modified version was not as sensitive for rapid cooling. This heuristic was named as Recursive SA (R-SA).

The R-SA utilizes SA’s idea of random walk when choosing expert sets. The core idea of the Recursive SA (R-SA) is to rollback to the best solution known after a fixed number of iterations without a move improving the objective function’s value. The solution where the R-SA makes rollback is probably bad because of series of bad local moves done in the iteration. The
rationale here is that it is faster to rollback to the best solution known than to wait the SA heuristic to correct its mistakes.

The R-SA heuristic did not actually work as planned, but regardless of that it gave very good (the best) results. See section 4.4 for more discussion about this topic.

Glover [12] mentions also about combining SA with Tabu search, but this approach was not tried.

2.4.7 Continuous optimization and rounding

The optimum of continuous optimization problem is better or equal to the optimum of otherwise identical problem with integer restrictions. The CES-heuristic was tested against results got from optimizing otherwise identical, but continuous problem. Continuous problems were solved with MATLAB’s Optimization toolbox function *fmincon* that uses Sequential Quadratic Programming method (SQP). See Fletcher[3] for an overview of SQP and Matlab’s documentation [4] for information about MATLAB’s SQP implementation.

Rounding the results of continuous optimization was tried also, but it performed badly. This was an expected result [14].

3 Algorithm testing

3.1 Approach

The CES-heuristic was tested against the comparison heuristics and algorithms presented in section 2.4. Heuristics’ accuracy was measured as percentage of the optimum/best result found. All other heuristics but brute-force method are approximate heuristics. Therefore, it was possible to optimize globally only those small problems that could be solved with brute-force method. On larger problems the comparisons had to be done against the

- Best result found by the different approximate heuristics
- Optimum of the equivalent continuous problem

3.2 Test data

Generated test data was used to test CES-heuristic's accuracy. Characteristics of test data were derived from the application’s real-world context. The following assumptions were used to generate test data:
- Experts’ scores over criteria follow Standard Normal distribution, which was thought to be good approximation for distribution of people’s skills. To simulate normal (discrete) grades, the scores were transformed to discrete-valued scale from 1 to 5. This seemed not to affect results in any way.

- Experts’ scores do not correlate with each other

- Criteria weights follow Standard normal distribution

For each test, experts’ scores, experts’ costs and criteria weights were given randomized values. The number of criteria was from 5-25.

CES-heuristic was tested against other exact or approximate heuristics with different sizes of problems. For each problem size, 100 different random problems were generated and results were compared. Table 2 below presents the type of tests performed.

<table>
<thead>
<tr>
<th>Comparison algorithm</th>
<th>Problem size</th>
<th># Of iterations</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brute force</td>
<td>6 – 17</td>
<td>40</td>
<td>In problems size of 20 experts only 20 iterations done due to lengthy computations</td>
</tr>
<tr>
<td>Monte-Carlo selection</td>
<td>25 – 500</td>
<td>40</td>
<td>Rather useless in problems large than 100 experts. Not included into the final statistics.</td>
</tr>
<tr>
<td>Stepwise selection/elimination</td>
<td>25 – 750</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Steepest-Ascent-Mildest-Descent</td>
<td>25 - 750</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Simulated annealing</td>
<td>25 – 750</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Recursive Simulated annealing</td>
<td>25 – 750</td>
<td>100</td>
<td>Modified version of the Simulated Annealing heuristic.</td>
</tr>
<tr>
<td>SQP</td>
<td>25 -150</td>
<td>100</td>
<td>Continuous problem</td>
</tr>
<tr>
<td>SQP (rounding)</td>
<td>25 -150</td>
<td>100</td>
<td>Rounding down from of the optimum of the equivalent continuous problem</td>
</tr>
</tbody>
</table>

Table 2. Tests performed
4 Results

The CES-heuristic showed to be accurate enough (better than 95% of the optimum) for reasonable sized problems (panel size larger than 5) and its performance in terms of computing time was excellent, even in the largest problems (2 seconds in average for problems with expert pool size of 500).

4.1 Accuracy

The CES heuristic gave results that were from 90% to 100% of the optimum. The CES heuristic’s accuracy got better in larger problems. The results can be considered good.

CES heuristic’s accuracy was measured by comparing it to results given by

- Brute-force –method; global optimum, but suitable for small problems only
- Monte-Carlo method; suitable for small problems only
- Other approximate heuristics
- SQP; Equivalent continuous problem

4.1.1 Small problems (expert pool < 20)

For problems size of 20 and smaller, it was possible to compare results given by the CES-heuristic to global optimum solved with brute-force method. In these problems, panel size varied from 3 to 15.

Table 3 shows summary results of the tests against global optimum in small problems. The average over all tests problems was 98.2% of the optimum and the worst result found was 89.7% of the optimum. The problem in which the CES heuristic gave its worst result was very small; there were only seven (7) experts in expert pool and panel size was three (3).
<table>
<thead>
<tr>
<th>Panel size</th>
<th>Avg (%)</th>
<th>Min (%)</th>
<th># of tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>97.3</td>
<td>89.7</td>
<td>57</td>
</tr>
<tr>
<td>4</td>
<td>97.7</td>
<td>90.7</td>
<td>74</td>
</tr>
<tr>
<td>5</td>
<td>97.9</td>
<td>92.7</td>
<td>80</td>
</tr>
<tr>
<td>6</td>
<td>98.0</td>
<td>94.5</td>
<td>57</td>
</tr>
<tr>
<td>7</td>
<td>98.4</td>
<td>94.1</td>
<td>59</td>
</tr>
<tr>
<td>8</td>
<td>98.6</td>
<td>96.3</td>
<td>43</td>
</tr>
<tr>
<td>9</td>
<td>98.9</td>
<td>97.4</td>
<td>36</td>
</tr>
<tr>
<td>10</td>
<td>98.9</td>
<td>97.3</td>
<td>25</td>
</tr>
<tr>
<td>11</td>
<td>99.0</td>
<td>97.4</td>
<td>22</td>
</tr>
<tr>
<td>12</td>
<td>98.8</td>
<td>97.9</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>99.4</td>
<td>98.8</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>99.5</td>
<td>99.1</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>100.0</td>
<td>100.0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3. Results given by CES heuristic in small problems

Figure 4 illustrates how the CES-heuristic performed for smaller problems. One can see how results got better in terms of both average and worst case as the size of expert panels in the test problems increased. In problems with panel size eight (8) or over the worst results given by the CES-heuristic was over 95%. Results over 95% of the optimum value can be considered good.
4.1.2 Large problems (expert pool > 20)

The results from smaller problems seemed promising due to fact that the CES-heuristic seemed to give more accurate results once problem size increased. To get more evidence about this, the CES-heuristic was tested against optimum of the continuous problem and results found by other heuristics.

At first, the CES was run against Monte-Carlo method with problems sizes of 25-500 experts. For each expert set size, 40 test problems were generated. For each test problem, $2^{20}$ expert set were tested chosen by Monte-Carlo method. The expert sets were selected as described in section 2.4.2.

Monte-Carlo method showed up to be unsuitable to be used as a comparison algorithm, because it did not give good results in larger problems. The CES-heuristic gave better results in average than Monte-Carlo method (see Table 4 and Figure 5) in problems larger than 30 experts. However, comparison between the CES and Monte-Carlo selection show how the CES heuristic gives notably better results than random selection of experts.

<table>
<thead>
<tr>
<th>Expert pool</th>
<th>25</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>75</th>
<th>100</th>
<th>125</th>
<th>150</th>
<th>200</th>
<th>250</th>
<th>300</th>
<th>400</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>CES/MC avg</td>
<td>99.9%</td>
<td>100.4%</td>
<td>101.4%</td>
<td>102.1%</td>
<td>103.2%</td>
<td>103.8%</td>
<td>104.5%</td>
<td>104.8%</td>
<td>105.8%</td>
<td>106.2%</td>
<td>107.0%</td>
<td>107.9%</td>
<td>108.3%</td>
</tr>
<tr>
<td>CES/MC lowest</td>
<td>97.4%</td>
<td>97.9%</td>
<td>98.8%</td>
<td>99.4%</td>
<td>100.4%</td>
<td>100.8%</td>
<td>101.3%</td>
<td>102.3%</td>
<td>103.2%</td>
<td>103.1%</td>
<td>102.4%</td>
<td>104.9%</td>
<td>105.5%</td>
</tr>
</tbody>
</table>

Table 4. CES-heuristic vs. Monte-Carlo selection
Figure 5. CES heuristic vs. Monte-Carlo method

The optimum of the equivalent continuous problem is the same or larger than the optimum of the discrete problem. When results given by the CES-heuristic were compared to the optimums of equivalent, but continuous problems, one could notice the same trend; results given by the CES-heuristic got better as the size of problem increased. When panel size was over 20, the CES-heuristic gave results 99% of the optimum of the equivalent continuous problem, at least. Real-world expert selection problems do not require any greater accuracy. On the other hand, panel size of 20 is quite large.
Figure 6. Results of CES-heuristic compared to continuous optimum

4.1.3 Comparison to between the approximate heuristics

According to statistical tests run SAMD, SA and Recursive-SA heuristics gave significantly better results than the CES heuristic. The difference in the objective function’s value was not large, but statistically significant.

The comparison of the approximate heuristics was done by comparing the ratio between heuristic’s results and the best result given by the heuristics.

\[ \text{test}_{ij} = \frac{\text{value}_{ij}}{\max_i (\text{value}_{ij})} \]  

(5)

where i marks for heuristic and j for test case).

Comparison statistics were calculated from data collected from 700 test problems (size of 25 to 750 experts). Average and worst result is shown in Table 5. The CES heuristic performed better than Stepwise heuristic and rounded SQP results, but worse than Simulated Annealing, Recursive Simulated annealing or SAMD-heuristics. Distribution of the results is shown in Figure 7. It can be seen that the R-SA heuristic gave better results than other heuristics and that
SAMD2 heuristic (comparison by incremental value) gave better results than SAMD3 heuristic (comparison by incremental value/cost ratio).

<table>
<thead>
<tr>
<th></th>
<th>CES</th>
<th>STEPWISE</th>
<th>SAMD2</th>
<th>SAMD3</th>
<th>SA2</th>
<th>R-SA</th>
<th>SQP (rounded)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst</td>
<td>92.88%</td>
<td>92.88%</td>
<td>96.57%</td>
<td>96.60%</td>
<td>98.08%</td>
<td>97.84%</td>
<td>92.88%</td>
</tr>
<tr>
<td>Average</td>
<td>99.00%</td>
<td>98.91%</td>
<td>99.85%</td>
<td>99.67%</td>
<td>99.77%</td>
<td>99.95%</td>
<td>98.94%</td>
</tr>
</tbody>
</table>

Table 5. Average of the best result found

Figure 7. Distribution of heuristic’s results (25 ≤ Ne ≤ 750)

To calculate statistical significance of the differences in the heuristics’ results, the test data was tested for normality first. According to Lilliefors test [4], the data was not from normal distribution (alpha = 0.01). Therefore, comparison statistics were calculated with nonparametric Kruskal-Wallis test [8]. Kruskal-Wallis test is a test for equal medians. Histogram (Figure 8) of the test data showed neither any signs of normality.

Because Matlab’s SQP implementation could solve only small- and medium-sized problems (expert pool size ≤ 150) in reasonable time and rounding SQP’s results gave bad results, rounded SQP was not included into the test statistics (25 ≤ expert pool size ≤ 750). It was known beforehand that rounding does not provide good results in general [14].
Figure 8. Histogram of the relative heuristic results

Low P-value (0.00) given by Kruskal-Wallis test (see Table 6) meant that there were significant differences between the heuristics. Heuristics were next compared with *multiple comparison procedure* to find out how heuristics differed from each other.

<table>
<thead>
<tr>
<th>Source</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>Chi-sq</th>
<th>Prob&gt;Chi-sq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns</td>
<td>2305590092.9393</td>
<td>5</td>
<td>461118018.5879</td>
<td>1719.6047</td>
<td>0</td>
</tr>
<tr>
<td>Error</td>
<td>3324292666.5607</td>
<td>4194</td>
<td>792630.5833</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>5629882759.5</td>
<td>4199</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6. Kruskal-Wallis ANOVA Table for heuristic results

According multiple comparison procedure results given by the CES- and stepwise- heuristics did not differ from each other significantly, but others did (see Figure 9). The heuristics’ order of superiority was therefore (from the most to accurate least accurate):

1. Recursive Simulated Annealing
2. Steepest-Ascent-Mildest-Descent using incremental value as comparison procedure
3. Simulated Annealing

4. Steepest-Ascent-Mildest-Descent using incremental value/cost ratio as comparison procedure

5. CES-heuristic and Stepwise selection procedure

**Figure 9. Multiple comparison procedure**

It must be noted that SA, SAMD and R-SA heuristics used CES-heuristic's result as initial value. Therefore these comparison heuristics could not give worse result than the CES heuristic.

### 4.2 Sensitivity analysis

The sensitivity of CES-heuristic's accuracy for number of criteria and size of expert panel was investigated.

#### 4.2.1 Sensitivity for number of criteria

Accuracy of the CES-heuristic seemed not to depend on the number of criteria. CES-heuristic's sensitivity on the number of criteria was tested by comparing results given by SQP (continuous problem) and CES-heuristic with different number of criteria. 40 tests were run for six (6) numbers of criteria (3, 5, 10, 15, 20, and 25). P-value 0,90 is a strong indicator for the heuristic's insensitivity to the number of criteria. Histogram plot (Figure 10) shows neither any sign of correlation between number of criteria and accuracy of the heuristic ($R^2 = 0.0021$).
### Kruskal-Wallis ANOVA Table

<table>
<thead>
<tr>
<th>Source</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>Chi-sq</th>
<th>Prob &gt; Chi-sq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns</td>
<td>7846.3</td>
<td>5</td>
<td>1569.26</td>
<td>1.6279</td>
<td>0.89786</td>
</tr>
<tr>
<td>Error</td>
<td>1144133.7</td>
<td>234</td>
<td>4889.4603</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1151980</td>
<td>239</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7. Kruskal-Wallis ANOVA Table for sensitivity over number of criteria

![Sensitivity for number of criteria](image)

Figure 10. CES-heuristic’s sensitivity on number of criteria

### 4.2.2 Sensitivity for size of the panel (budget)

The CES-heuristic gave more accurate results when the size of expert panel was increased.

To test CES-heuristic's sensitivity for panel size, 240 different-sized test problems were generated and variations in CES-heuristic's accuracy were measured. Accuracy was measured by comparing CES-heuristic's results to results of SQP (continuous problem). Because the distribution of expert's costs was kept same for all the tests, the size of budget was chosen as a parameter for testing CES-heuristic's sensitivity for panel size. Histogram (Figure 11) indicates non-linear correlation between budget (~panel size) and CES-heuristic's accuracy; therefore
Spearman Rank correlation [16] was used instead of Pearson correlation. Pearson correlation expects linear correlation.

\[ r' = 1 - 6 \sum \frac{d^2}{N(N^2 - 1)} \]  \hspace{1cm} (6)

where \( d \) is the difference in rank of corresponding variables.

Calculated correlation coefficient \( r' \) was 0.3379, which was significance tested with \( H_0: r' = 0, H_1: r' \neq 0 \). For large samples P-value can be calculated from standard normal distribution approximation [17]:

\[ Z = r' \sqrt{\frac{N-1}{N}} \]  \hspace{1cm} (7)

P-value was \( 1.75 \times 10^{-7} \) so \( H_0 \) was abandoned. According to test data there is significant and positive correlation between accuracy and panel size.

This result is common sense. Because the CES-heuristic chooses experts according to cost/benefit-ratio, it is in one sense blind to budget constraint. Once panel size is increased the

Figure 11. CES-heuristic’s sensitivity for panel size

The Spearman rank correlation coefficient is defined by

\[ r' \equiv 1 - 6 \sum \frac{d^2}{N(N^2 - 1)} \]  \hspace{1cm} (6)

where \( d \) is the difference in rank of corresponding variables.

Calculated correlation coefficient \( r' \) was 0.3379, which was significance tested with \( H_0: r' = 0, H_1: r' \neq 0 \). For large samples P-value can be calculated from standard normal distribution approximation [17]:

\[ Z = r' \sqrt{\frac{N-1}{N}} \]  \hspace{1cm} (7)

P-value was \( 1.75 \times 10^{-7} \) so \( H_0 \) was abandoned. According to test data there is significant and positive correlation between accuracy and panel size.

This result is common sense. Because the CES-heuristic chooses experts according to cost/benefit-ratio, it is in one sense blind to budget constraint. Once panel size is increased the
effect of an individual expert selection (usually the last expert chosen) diminishes. Concave objective function should also reduce the difference between average and optimum solution as budget constraint is increased.

4.3 Computing efficiency

The performance of the CES-heuristic showed to be very good in terms of computation time, even in very large problems.

The complexity of the expert selection problem increases in $2^{\text{size of expert pool}}$. Time required searching optimum using brute-force method doubles every time an expert is added to expert pool (see Figure 12). However, the CES-heuristic was able to solve a problem size of 2500 experts (and 20 experts in panel) in 45 seconds with normal PC-hardware (Intel Pentium III 650 MHz). No other comparison heuristic could match CES-heuristic’s performance.

![Problem complexity graph](image)

**Figure 12. Problem complexity**

The computation time of the CES-heuristic follows approximately the equation given below.

$$t_{CES} \approx A \cdot n^2 \cdot p \cdot c^2$$

Where $A$ is some computer- and software -specific factor, $n$ is size of expert pool, $p$ is size of expert panel and $c$ is number of criteria.

The computation time of Cost-inefficiency elimination follows approximately
\[ t_{CIE} \approx A \cdot n^2 \cdot (n - p) \cdot c^2 \quad (9) \]
\[ t_{CIE} \approx A \cdot n^3 \cdot c^2 \quad \text{if } n \gg p \quad (10) \]

Figure 13 and Table 8 show how the heuristics performed in average. The computation time averages have been calculated from 100 iterations for each problem size. The sizes of the test problems varied from 25 to 750 with the approximate heuristics and from 20 to 150 with SQP. There were 5 criteria in each test problem and the sizes of expert panels varied between 4 and 14. Note that the time-axis has logarithmic scale in the figures.

SQP did not scale as well as the approximate heuristics. The computing time of the SA and R-SA heuristics depended on parameter settings used. Test heuristic's parameters were chosen with the way the heuristics give best possible results in reasonable time. With faster settings, stochastic heuristics (SA and R-SA) gave worse results (i.e. were more often unable to improve initial solution).
Figure 13. Average heuristic performance

<table>
<thead>
<tr>
<th>Ne</th>
<th>CES</th>
<th>STEPWISE</th>
<th>SAMD2</th>
<th>SAMD3</th>
<th>SA2</th>
<th>R-SA</th>
<th>SQP</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>0.1</td>
<td>0.1</td>
<td>2.7</td>
<td>2.0</td>
<td>63.0</td>
<td>26.5</td>
<td>2.6</td>
</tr>
<tr>
<td>150</td>
<td>0.4</td>
<td>0.9</td>
<td>16.6</td>
<td>9.8</td>
<td>238.9</td>
<td>78.8</td>
<td>503.5</td>
</tr>
<tr>
<td>750</td>
<td>3.3</td>
<td>5.8</td>
<td>98.8</td>
<td>70.5</td>
<td>118.3</td>
<td>398.9</td>
<td></td>
</tr>
</tbody>
</table>

Table 8. Average heuristic performance in seconds (Ne = expert pool size)

4.4 Analysis of the results

If one takes account how precise the input data in the expert selection problem can be at its best, the CES-heuristic can be considered to give results that are accurate enough for real-life use. By looking at Figure 14 one can notice that the CES heuristic gives notably better results than random selection of expert sets. The CES-heuristic also performed very well and its performance was satisfactory for real-world applications even with running at standard PC-hardware (650 MHz).
Figure 14. CES-heuristic compared to random sample

Anyhow, CES-heuristic’s propensity to stop after the first local optima found, makes it unsuitable to be used for general nonlinear 0-1 programming problems. Many researchers have found SAMD to perform better than SA and in sense of accuracy and computing time. In this study SAMD performed the second best and was very computing efficient. From the heuristics tested, author regards SAMD as the most promising heuristic for general nonlinear 0-1 programming problems.

The Recursive Simulated Annealing (R-SA) heuristic gave the best results, but more careful study is needed before R-SA can be considered as a strong nonlinear 0-1 programming heuristic. Closer look to R-SA’s functioning revealed that it did not work as planned (R-SA heuristic is presented in appendix 1). In practice the R-SA heuristic worked with the following way:

1. Do fixed number of random moves accepted by SA’s local move criterion. Set \textit{base solution} = current expert.
2. Loop fixed number of random one-point moves. Make the moves from the base solution.
3. If improvement found
   a. Save it as \textit{best solution}
   b. Go to step 5.
4. No improvement found,
   a. Lower temperature: \textit{TEMP} = \textit{TEMP} \times \text{COOLING};
   b. Set base solution = \textit{base solution} + random move
   c. Go to step 5.
5. if \texttt{TEMP > TEMP\_LIMIT} go back to step 2. if not terminate heuristic and return best solution found

Instead of making local moves (at step 2) for certain number of rounds and then doing rollback to the best solution found, the heuristic did only moves that improved the solution. Because local moves were not done at depth == DEPTH\_LIMIT, all attempts were basically one or two moves from the \textit{base solution}. The R-SA heuristic did not actually move far away from local optima, but tried to found "remove and add"-moves near local optima that improved solution. Author’s interpretation is that the R-SA improved solution by higher utilization of the available budget instead of finding solutions that were fundamentally different from the solution given by the CES heuristic.

This finding is very interesting. It gives reason to suspect that one basic design choice was done wrong with all the approximate heuristics implemented. All the approximate heuristics were implemented with the way that one move meant either adding or removing one expert. Another approach would have been to compose one move from two half-moves (1 move = remove expert + add expert). Some of the heuristics (SAMD) would have been more complicated to implement this way, but it might have improved the results given by SA- and SAMD-heuristics. Glover discusses about this approach in his article [12].

A simple experiment was done with heuristic that improves initial solution given by the CES-heuristic by making the best improving remove & add –moves. A test statistics from 180 generated test problems gave promising results. Only the R-SA heuristic gave slightly better results than this simple and very fast heuristic.

Re-implementing the SA- and SAMD-heuristics with this \textit{2-half-moves-structure} was seen as the most promising direction for future research, but the re-implementation was considered to be out of the scope of this study.

5 Real-world applicability and final conclusions

The CES-heuristic showed to be accurate enough for the formulated expert selection problem and it was found to be very computing efficient. Its propensity to stop after first local optima found makes it unsuitable for solving general nonlinear 0-1 programming problems. The CES-heuristic or \textit{steepest-ascent-one-point-move} heuristic as it is called in research literature is not regarded as strong heuristic by researchers [11].

Question more difficult to answer is that is it, or is the whole expert selection problem formulation valid for real-world use?
To work, the heuristic requires expert pool evaluated according to common criteria. At first, a question rises about whether it is realistic to expect that criteria and scores can be chosen in a valid way? Even if common quantitative criteria are known (which is quite a assumption), can the initiator of the technology foresight panel state weights for criteria? Or can the panel initiator understand how criteria weights affect to the objective function? Objective function’s – which was formulated as a sum of concave utility functions (In(1+x), for example) – sensitivity for criteria weights is not trivial. At least significant support must be provided to the person deciding criteria and criteria weights.

Secondly, the assumption that such quantitative data exists that can be used for evaluating experts and the assumption that the quantitative data is sufficient for expert selection has no evidence. It is true that mathematical formula of panel selection is objective and honest, but the formulation of the formula and its weights is not. If it is question about national decision-making, there is always political intent to affect the panel’s composition. Technology strategy formulation is not scientific, but political process. By affecting to the early stages of the process (conclusions given by technology foresight panel) one can affect the best to the final decisions.

Thirdly, one can also argue that the real problem is to find relevant experts (expert pool) than choose a set of experts among them. The CES-heuristic requires set of experts evaluated against common criteria, but it does not help one at all to find the relevant experts.

Fourthly, The CES-heuristic does not take panel members’ personalities and personality matches into account. How much this counts depends on how technology foresight panel is organized. If the panel members should work as a team to formulate common view on the subject, it can be beneficial to take personality matches into account. At the field of personnel selection research, [18] has made research about team members’ personalities in successful product development teams. One of the [18]’s findings was that there is strong and significant negative correlation between the heterogeneity of team performance and team members’ conscientiousness (the degree to which a person is dependable, purposeful and strong-willed). The expert selection formulation presented in this paper does not consider dependencies between panel members at all. Anyhow, if technology foresight panel is arranged as Delphi study or similar, this property has less significance.

Without better information about expert panel selection, the applicability of the CES-heuristic remained open. CES-heuristic is good heuristic to solve the formulated mathematical problem, but is the problem really the right problem that needs to be solved? Considering the points raised above, author’s judgment is that CES-heuristic does not help in real-world expert selection problem.
## References

<table>
<thead>
<tr>
<th>#Ref</th>
<th>Description</th>
</tr>
</thead>
</table>
| [1]  | Nonlinear programming FAQ,  
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| [16] | Spearman Rank Correlation Coefficient.  
| [17] | IFA services, Institute of phonetic sciences, Universiteit van Amsterdam,  
6 Appendices

6.1 Recursive Simulated Annealing

Author named his modification to Simulated Annealing as Recursive Simulated Annealing. The purpose of the modification was to enable doing rollback to best-known solution if after certain amount of iteration rounds if no improvement is found. Rationale behind this algorithm is that SA may do bad local moves at early stages when the probability of acceptance is high. There is no any great theory behind this approach, but it just performed very well in tests run. Closer investigation revealed that good performance might have come from the heuristic's propensity to find good add and remove expert -combinations.

```
Function expert_set = recursiveSA(sa_Experts, best_experts, depth)
Global TEMP = TEMP_START; % SA algorithm uses the same temperature variable
Change = TRUE;

While (change & TEMP > TEMP_LIMIT)
    Change = FALSE;
    For i = 1:REP
        Tmp_experts = makeRandomChange(sa_experts);

        DValue = value(tmp_experts) - value(best_expert_set);
        If Dvalue > 0
            Return tmp_experts;
        Elseif ( (depth < DEPTH_LIMIT) & (rand() < Exp(Dvalue/TEMP) )
            Sa_experts = recursiveSA(tmp_experts, best_experts, depth+1);
            If value(sa_experts) > value(best_expert_set);
                Return sa_experts;
        End;
    End;
End; % while

Return best_expert_set;
```

Equation 2. Simulated Annealing algorithm with periodic rollback