

Using Genetic Algorithms for Investigating Specific Regions of the Solution Space

Irfan Younas[†], Rassul Ayani[†] and Johan Schubert[‡]

[†]School of Information and Communication Technology
KTH Royal Institute of Technology, Stockholm, Sweden
{irfany, ayani}@kth.se

[‡]Swedish Defence Research Agency, Stockholm, Sweden
schubert@foi.se

Abstract—In many practical cases decision makers are interested to understand the whole solution space, including possible outliers. By outlier we mean there is a solution that is theoretically possible, though with very low probability that it occurs. In many combinatorial problems, this is a very challenging task. During the past decade, the Data Farming community has done substantial work on developing methods and techniques for better understanding of the solution space. The data farming community has also looked at the design of experiments and used Latin hypercube (LH) techniques for this purpose. The LH is proven to be one of the important sampling methods for selecting a representative subset of the input space.

In this paper, we consider a company X that wants to outsource m subprojects of a given project P . We assume that there are n potential subcontractors for each subproject. Thus, there will be n^m ways to assign the subprojects to the potential subcontractors. The project manager is interested to find those assignments that complete the project within a given time and a given cost frame. An exhaustive examination of all assignments is not feasible, if m and n are big numbers. We propose an objective-based genetic algorithm (GA) for finding the set of assignments that are mapped onto a given subset of the solution space. It means, as opposed to the design of experiment techniques, we start from the solution space and try to find the combinations of the input parameter values that can lead to a specific region of the solution space. By some numerical examples, we show how our GA identifies the set of such feasible assignments.

Key words: Data Farming, Design of Experiments, Assignment Problem, Genetic Algorithms, Evolutionary Algorithms, Decision-support Systems, Project Management

I. INTRODUCTION

There are often many parameters (factors) which can have direct or indirect impact on the decisions we make. The problem of how to make intelligent and most promising decisions considering all the underlying factors and conditions is an important research topic. Searching for affects by varying one parameter at a time is an ineffective and usually inaccurate means of getting insights and understanding of the underlying system.

In such combinatorial problems, we denote the set of all combinations of the input parameter values as input space, and the set of all possible outcomes as solution space. In many situations, the decision makers are interested to have a good understanding of the solution space, i.e., to understand what are the potential solutions of the problem, including

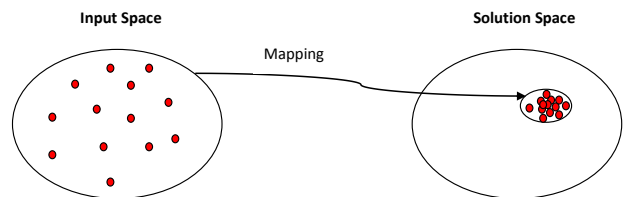


Fig. 1. Illustrates the subset of the inputs space (denoted by red points) that are mapped to a given region of the solutions space

possible outliers. By outlier we mean there is a solution that is theoretically possible, though with very low probability that it occurs. For instance, consider planning the time and the cost needed to complete a project P . Further, assume that P consists of m subprojects p_1 to p_m and there are n different ways to do each subproject, e.g., there are n different subcontractors competing for each subproject. This will result in n^m different ways of completing P , possibly with different costs and time requirements. If m and n are big numbers, then it is inefficient, and sometimes almost impossible, to consider all these combinations, e.g., if $m = 100$ and $n = 10$, then there are 10^{100} combinations of the input values.

In the past decade, the Data Farming community has put a lot of efforts in investigating how to get a better understanding of the whole solution space, including possible outliers [1]–[3]. The data farming community has also looked at the design of experiments. There has been a substantial amount of work on design of experiments, where the focus is on the input space.

Latin hypercube (LH) sampling [4] has proven to be “an invaluable technique” [5] and it is one of the most widely used and popular methods for designing experiments [6]. The objective of the LH is to select an “adequate” subset of the input space. Extensive work has been done on a wide variation of LHs with desirable properties such as orthogonality and space-filling. In order to generate orthogonal (no correlation between input variables), nearly orthogonal and space-filling designs, various researchers like Ye [7], Cioppa [8], [9], Joshua [6], Steinberg et al. [10], and Hernandez [5] have contributed to this area.

Most of the LH methods give an adequate representation of

the input space. However, it is unclear how well the selected subset of the input space is mapped onto the solution space. For instance, one may imagine the cases where the selected subset of the input space is mapped to a small region of the solution space as shown in Figure 1.

On the other hand, the main objective of this work is to contribute to better understanding of the solution space. We propose an objective-based genetic algorithm (GA) for finding the set of input data that are mapped onto a given subset of the solution space. It means, as opposed to the design of experiment techniques, we try to answer to the following question: Is there any combination of the input parameter values that can lead to a solution within a given subset of the solution space? For instance in case of projects outsourcing, our objective could be to select the subcontractors such that the cost is minimized and the time threshold is satisfied. By varying the time threshold, we can find different economical, feasible and cost efficient options.

The outline of the rest of this paper is as follows. In Section II, a mathematical formulation of the problem is presented. In section III, the proposed GA heuristic is described. Section IV presents the experimental results. Section V discusses further applications of our GA. Finally we conclude this paper in section VI.

II. PROBLEM FORMULATION

In this paper, as an example, we consider a specific application where a company X wants to outsource m subprojects, p_1 to p_m of a given project P . For each subproject, there are n potential subcontractors. The proposed cost and time needed to complete a subproject proposed by different subcontractors generally differ. This means that for each subproject p_i we have n potential subcontractors and each of them contains a cost and the completion time.

The objective is to find the best selection of the potential subcontractors such that the cost is minimized and the total time needed to complete all subprojects is less than a given threshold. Thus the m subprojects are represented by m factors and the n potential subcontractors by n levels. We define $c(p_i, l_j)$ and $t(p_i, l_j)$ as the cost and time of performing project p_i by subcontractor l_j respectively. We also assume the dependency between subprojects by a graph similar to Figure 2. The subprojects are represented by nodes on graph $\mathcal{G} = \{V, E\}$, where $V = \mathcal{P} \cup \{S, T\}$ is the set of nodes representing subprojects and E is the set of directed edges (arcs) representing the dependency (precedence constraints) of subprojects. Here S and T are starting and terminating nodes.

For this scenario Optimality is defined as minimizing the objective function (the cost function)

$$y(\mathcal{X}) = \sum_{i=1}^m \sum_{j=1}^n c(p_i, l_j) x_{ij}, \quad (1)$$

(Here $\mathcal{X} = [x_{ij}]_{m \times n}$ is an *assignment matrix* where $x_{ij} = 1$ if level l_j is assigned to factor p_i and 0 otherwise.)

subject to constraints

$$\sum_{j=1}^n x_{ij} = 1, \quad \forall i | p_i \in \mathcal{P} \quad (2)$$

$$t_{m+2}^{cp} \leq T, \quad (3)$$

where

$$t_i^{cp} = \max_{(k,i) \in E} \{t_k^{cp} + \sum_{j=1}^n t(p_i, l_j) x_{ij}\}, \quad \forall i \in V \setminus \{S\} \quad (4)$$

in Topological (linearized) order

$$x_{ij} \in \{0, 1\} \quad \forall i | p_i \in \mathcal{P}, l_j \in \mathcal{L} \quad (5)$$

Constraint (2) ensures that each factor is assigned to one level and constraint (3) defines the feasible region of the search space. Constraint (3) shows that time for critical path (t_{m+2}^{cp} is the time of the longest path from starting node S to terminating node T) of a solution should be less than or equal to the defined time threshold. $\mathcal{T}^{cp} = \{t_1^{cp}, \dots, t_{m+2}^{cp}\}$ is the vector for maintaining times (accumulative of all predecessors) of $m+2$ vertices (nodes).

III. THE GA HEURISTIC

Genetic algorithms (GAs) [11], [12] are famous meta heuristics which are inspired by evolutionary ideas of genetics and natural selection [13]. GAs may be adopted using specific features of an application or by using a general Evolutionary Computing Modeling Language (ECML) proposed in [14]. In this paper, we develop a GA tailored to the project management problem. Our GA works with population of individuals which are called potential solutions. Each potential solution consists of a set of genes and the collection of these genes form a chromosome. Each chromosome in GA represents one candidate solution for the problem. The GAs evolve the population of candidate solutions. The first population is generated at random and the fitness value is calculated according to given fitness function. In each iteration some of the promising solutions are selected from the population of candidate solutions. New candidate solutions are created by applying crossover and mutation to the chosen solutions. Due to crossover, the generated solutions inherit characteristics from both parents. Mutation prevents the loss of diversity [13] and is helpful to traverse different regions of search space to locate hidden solutions by escaping local minima/maxima. The steps involved in our algorithm are as follows:

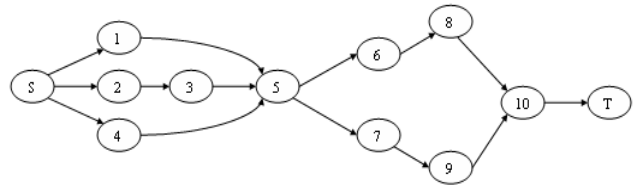


Fig. 2. Dependency Graph for subprojects

| | | | | | | | | | | |
|---------|---|---|---|---|---|---|---|---|---|----|
| Factors | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Levels | 3 | 5 | 3 | 2 | 4 | 5 | 2 | 6 | 2 | 5 |

Fig. 3. Chromosome representation of a candidate solution

- 1) First step is to devise a suitable representation scheme for candidate solutions (which are called chromosomes in GA literature). In our problem, the set of given factors is represented by a m -dimensional vector of integer values where each factor $p_i \in \mathcal{P}$ takes a level $l_j \in \mathcal{L}$. For example, consider a case where we have 10 factors and we assume that there are 6 levels for each factor. Figure 3 shows a possible assignment of n levels to m factors where $m = 10$ and $n = 6$ and Figure 2 shows the dependency graph of m factors. An initial population with M distinct candidate solutions (sample points) is constructed by randomly assigning levels to factors. The identical solutions can be cause of early convergence to local minimum and we may not be able to find the global optimal and other hidden solutions.
- 2) In order to evaluate the suitability of a candidate solution, fitness and unfitness values are calculated as done in [15]. For the problem discussed in section II, our objective function is to minimize the total cost of the levels assigned to factors which is the fitness value. The unfitness value is related to the violation of constraints and it is a measure of infeasibility. Constraint (3) ensures that the total time required to complete all the subprojects should not exceed the defined threshold T . Unfitness value u_s of a solution s is calculated in similar way as done by Chu et al. [15] and is given by

$$u_s = \max[0, t_{m+2}^{cp} - T] \quad (6)$$

where t_{m+2}^{cp} is the critical path (longest path with respect to projects completion time which is the time necessary to complete all the subprojects) which is represented by equation (3) and (4) and is calculated using dynamic programming [16], [17] as shown in Figure 4.

We can see from equation (6), that for a feasible solution s , unfitness value $u_s = 0$. After calculating the fitness and unfitness values, the population of individuals is sorted based on two values (unfitness and fitness). For feasible solutions ($u_s = 0$), the candidate solutions with lower cost are said to be more suitable (fit). On the other hand, solutions with highest unfitness value are the weakest individuals.

- 3) Selection criteria is applied to choose two potential solutions for reproduction from the population. There are different selection schemes as discussed by Goldberg et al. [18]. We choose binary tournament selection which was also used in [15] where two individuals are selected randomly with same likelihood of being selected. The fitter individual is allocated for reproduction trial. For reproduction two parents are chosen by conducting two

Algorithm for longest path directed acyclic graph

Input: graph G , chromosome genes, vertices of G in topological order

Output: LongestPath (minimum time required)

$|V|$ = Set of vertices of graph G

distTo = Array with $|V(G)|$ integer elements initialized to $-\infty$

distTo(s) = 0

for each vertex $v \in V$ in topological order **do**

for each edge $(v,u) \in E$ **do**

$t(v,u)$ = time weight of v (completion time of node (project) v)

If distTo(u) < distTo(v) + $t(v,u)$ **then**
distTo(u) = distTo(v) + $t(v,u)$

LongestPath = max (distTo(v) for all v in $|V|$)

Return LongestPath

Fig. 4. Algorithm: Longest Path of projects dependency graph (minimum time necessary to complete all the projects)

independent binary tournaments.

- 4) Two children are generated by applying crossover operator to the chosen parent solutions. There are different crossover operators. We choose one-point crossover in which the crossover point p is randomly selected between 1 and m . Child A takes first p genes from parent X and the remaining $m - p$ genes from parent Y while child B takes first p genes from parent Y and the remaining $m - p$ genes from parent X. A child in this way inherits features from both parents and thus represents a different point in the input space. The ultimate objective of the GA is to transfer the characteristics from stronger individuals to their children from generation to generation and evolve the population according to the defined objective function.
- 5) Generally, the GA can be trapped by local minimum/maximum. In order to escape from these local optimums and to explore different regions of the search space, we use mutation operator. There are different mutation operators which are selected according to representation scheme of individuals. For our problem we select simple mutation operator in which one of the factor is assigned a random level $l_j \in \mathcal{L}$ with some defined mutation probability p . After crossover and mutation the corresponding child solution can be infeasible (for example in our case the time for completing all tasks can exceed total threshold T). Sorting of chromosomes is done with respect to fitness and unfitness value and the solutions which are infeasible they have unfitness value > 0 and they are considered poor solutions.
- 6) In order to evolve the population keeping it constant throughout the process, replacement scheme is applied to replace an individual in the population with the generated children chromosomes. In our approach the weakest (worst) individual in the population is replaced

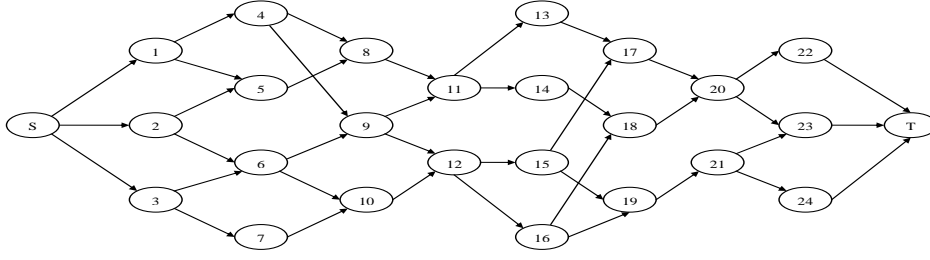


Fig. 5. Dependency Graph of subProjects

by the child solution.

- 7) The iterative process (from steps 3 to 6) is repeated until some termination criteria or condition for solution quality is satisfied. For example we say that the algorithm should terminate when we have generated N non-duplicate children without improving the best candidate solution found so far.

On termination of the algorithm, we have a better (stronger) population of individuals than the initial population. By analyzing these candidate solutions, decision makers and analysts can find different options which are more flexible, feasible and matching to their interests.

IV. EXPERIMENTAL RESULTS

In this section, we present a set of experiments and the obtained results. We consider a company X that wants to outsource its m subprojects, where there are n subcontractors for each of them. For our experiments we assume that $m = 24$ and $n = 5$ and the dependencies of the subprojects are shown by a graph $\mathcal{G} = \{V, E\}$ (see Figure 5), where $V = \mathcal{P} \cup \{S, T\}$ is a set of nodes representing subprojects and E is the set of directed edges (arcs) representing the dependency (precedence constraints) of subprojects. In the Figure 5, S and T are the starting and terminating nodes. A directed arc $(p_i, p_k) \in E$ depicts that the subproject p_k is dependent on subproject p_i , i.e., p_i must be completed before p_k is started.

For input data, we generate synthetic cost and completion time of each subproject. We denote the estimated cost of subproject p_i by c_i and generate it as a random number within $[100, 400]$. Similarly, the estimated completion time of p_i is denoted by t_i and is generated as a random number within $[100, 200]$. Once c_i and t_i are estimated we generate 5 pairs (c_{i1}, t_{i1}) , (c_{i2}, t_{i2}) , (c_{i3}, t_{i3}) , (c_{i4}, t_{i4}) and (c_{i5}, t_{i5}) corresponding to the cost and time of the five offers submitted by the five potential subcontractors for p_i (where $c_{ij} = c(p_i, l_j)$ and $t_{ij} = t(p_i, l_j)$). We assume that $0.5c_i \leq c_{ij} \leq 1.5c_i$ and $0.8t_i \leq t_{ij} \leq 1.2t_i$, $\forall i \in \{1, \dots, m\}$ and $\forall j \in \{1, \dots, n\}$.

Now we can identify the solution space by determining:

- The minimum time needed for completing all subprojects, T_{min} .
- The minimum cost required for completing all subprojects, C_{min} .
- The maximum time needed for completing all subprojects, T_{max} .

- The maximum cost for completing all subprojects, C_{max} .

T_{min} (and T_{max}) can be calculated by selecting the subcontractor with the lowest (and highest) completion time for each subproject (without taking into account the cost) and sum up the time for nodes on the critical path of the corresponding graph i.e.,

$$T_{min} = t_{m+2}^{cp}, \quad (7)$$

where

$$t_i^{cp} = \max_{(k,i) \in E} \{t_k^{cp} + \min_{1 \leq j \leq n} \{t_{ij}\}\}, \quad \forall i \in V \setminus \{S\} \quad (8)$$

$$T_{max} = t_{m+2}^{cp}, \quad (9)$$

where

$$t_i^{cp} = \max_{(k,i) \in E} \{t_k^{cp} + \max_{1 \leq j \leq n} \{t_{ij}\}\}, \quad \forall i \in V \setminus \{S\} \quad (10)$$

Similarly, C_{min} (and C_{max}) can be easily calculated by selecting the subcontractor with the lowest (and highest) proposed cost for each subproject, without taking into account the completion times, i.e.,

$$C_{min} = \sum_{i=1}^m \min_{1 \leq j \leq n} \{c_{ij}\} \quad (11)$$

$$C_{max} = \sum_{i=1}^m \max_{1 \leq j \leq n} \{c_{ij}\} \quad (12)$$

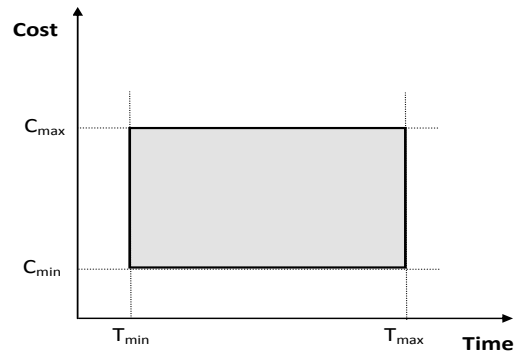


Fig. 6. Solution Space (bordered by rectangle)

TABLE I
20 DESIGN POINTS USING NEARLY ORTHOGONAL AND SPACE FILLING LATIN HYPERCUBES

| Individual # | Total cost | Total time | Possible Assignment of subcontractors to subprojects (24-dimensional vector represents subprojects and the integer values denote the subcontractor assigned to each subproject) |
|--------------|------------|------------|---|
| 1 | 4714 | 1515 | [3, 3, 5, 5, 3, 1, 5, 2, 3, 4, 1, 1, 3, 2, 5, 4, 3, 3, 3, 1, 2, 4, 1, 4] |
| 2 | 4933 | 1423 | [3, 4, 2, 2, 2, 3, 4, 1, 3, 5, 2, 4, 3, 2, 5, 4, 3, 1, 4, 5, 1, 5, 3, 4] |
| 3 | 5063 | 1375 | [4, 3, 3, 3, 4, 3, 2, 4, 3, 2, 5, 1, 3, 5, 3, 1, 2, 5, 5, 2, 3, 1, 3, 3] |
| 4 | 5074 | 1438 | [3, 3, 5, 5, 4, 3, 3, 4, 1, 4, 2, 3, 4, 5, 1, 4, 2, 2, 4, 3, 5, 4, 2, 2] |
| 5 | 5094 | 1455 | [2, 5, 3, 5, 5, 1, 3, 5, 3, 5, 4, 3, 3, 5, 3, 3, 2, 3, 1, 1, 4, 4, 3] |
| 6 | 5126 | 1451 | [1, 5, 5, 5, 4, 4, 3, 3, 3, 4, 4, 1, 1, 5, 5, 4, 3, 2, 1, 4, 3, 3, 3, 1] |
| 7 | 5202 | 1397 | [3, 5, 4, 1, 5, 4, 5, 4, 4, 2, 1, 1, 3, 5, 4, 1, 1, 3, 4, 2, 1, 5, 5, 3] |
| 8 | 5207 | 1449 | [2, 5, 2, 3, 5, 4, 5, 2, 3, 5, 4, 3, 2, 5, 3, 5, 3, 2, 3, 1, 4, 2, 5, 3] |
| 9 | 5242 | 1434 | [1, 5, 2, 3, 1, 3, 1, 4, 1, 1, 5, 3, 3, 5, 3, 5, 2, 3, 4, 2, 1, 4, 1, 3] |
| 10 | 5249 | 1381 | [5, 5, 2, 2, 4, 2, 3, 4, 2, 4, 1, 1, 2, 2, 2, 2, 4, 1, 3, 1, 3, 2, 1, 4] |
| 11 | 5284 | 1407 | [2, 5, 4, 2, 3, 4, 5, 3, 2, 1, 1, 2, 2, 1, 5, 2, 3, 5, 5, 3, 1, 1, 3, 4] |
| 12 | 5299 | 1436 | [5, 3, 2, 2, 5, 4, 1, 4, 1, 4, 5, 4, 2, 3, 2, 1, 5, 1, 4, 4, 1, 4, 1, 3] |
| 13 | 5304 | 1355 | [2, 3, 3, 4, 1, 2, 3, 2, 1, 4, 1, 5, 3, 5, 4, 3, 1, 5, 5, 4, 3, 3, 5, 4] |
| 14 | 5337 | 1409 | [5, 2, 2, 2, 3, 3, 2, 1, 4, 1, 2, 2, 4, 5, 2, 5, 3, 3, 5, 4, 3, 1, 5, 3] |
| 15 | 5340 | 1368 | [3, 3, 2, 3, 3, 2, 4, 2, 5, 4, 4, 2, 2, 3, 2, 2, 5, 3, 5, 4, 4, 5, 3, 3] |
| 16 | 5340 | 1449 | [1, 5, 3, 3, 5, 4, 1, 4, 5, 4, 4, 1, 3, 5, 5, 2, 5, 2, 2, 4, 2, 4, 5, 4] |
| 17 | 5345 | 1423 | [5, 5, 4, 5, 4, 4, 4, 5, 1, 1, 4, 3, 4, 1, 5, 4, 4, 5, 4, 4, 1, 2, 1, 2] |
| 18 | 5349 | 1475 | [2, 3, 3, 4, 2, 4, 3, 2, 3, 4, 2, 3, 4, 1, 4, 2, 4, 5, 1, 5, 3, 3, 5, 4] |
| 19 | 5351 | 1438 | [5, 2, 3, 2, 1, 5, 5, 1, 1, 3, 4, 4, 4, 5, 3, 4, 3, 5, 5, 1, 3, 5, 1, 3] |
| 20 | 5351 | 1441 | [4, 3, 2, 5, 5, 4, 3, 4, 1, 5, 4, 2, 2, 5, 5, 2, 2, 3, 5, 1, 2, 5, 5, 2] |

Using equations (7), (9), (11) and (12) we obtain $T_{min} = 1192$, $T_{max} = 1604$, $C_{min} = 3575$ and $C_{max} = 7773$ and the expected solution space is the rectangle shown in Figure 6.

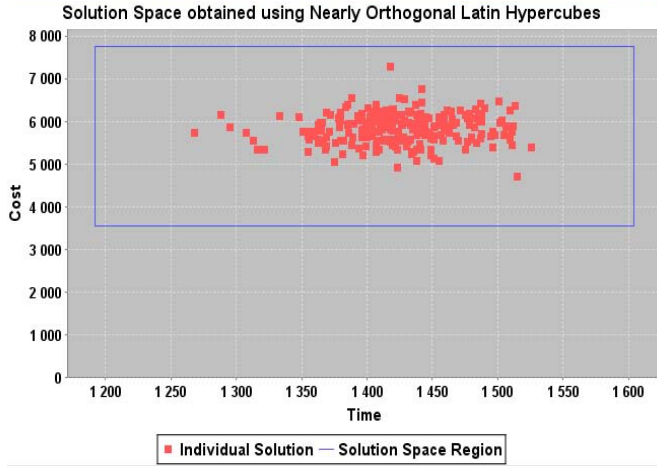


Fig. 7. Representation of the solution space using Nearly Orthogonal and space-filling Latin Hypercubes

First we select 257 points of the input space using nearly orthogonal and space-filling Latin hypercubes (NOLH), suggested by Cioppa and Lucas [9]. Each point is represented by a vector assigning each subproject to one of n potential subcontractors. For readability reason we present 20 of these design points in table I where each row represents a design point (or an assignment of subprojects to the subcontractors). Figure 7 shows how the 257 points are mapped onto the solution space, i.e., the cost and time of each of these assignments. The generated design points (or assignment vectors) using NOLH is shown to be a good representative of the input

space, since it fulfills the near orthogonality and space filling requirements [9].

On the other hand, we may be interested to know which design points are mapped into a specific region of the solution space. For instance, in our outsourcing example the project manager may raise the following questions:

- Which are the assignment vectors that complete the project with the minimum cost C_{min} ?
- Which are the assignment vectors that complete the project in T_{min} ?
- Can we identify the assignment vectors that complete the project within a given time threshold T ? If so, then list at most 20 of them with the lowest total cost?

Using our GA we obtain the answer to (a), (b) and (c) that are represented in Tables II, III and IV respectively. Our GA finds 4 design points that complete the project with the minimum cost, i.e., the total cost= 3575 (see Table II). Table III, illustrates 20 of the assignment vectors that fulfill the time constraint of question (b), i.e., the completion time is equal to 1192. Actually, there are more assignment vectors fulfilling this requirement, but for readability reason we present only the best 20, namely those that fulfill the time constraint but have the lowest cost. Similarly, Table IV illustrates 20 of the assignment vectors that complete the whole project in 1380 time units. In tables II, III and IV, the bold figures in each row identify the critical path of Figure 5 for that particular assignment.

We implemented our GA in Java and run it on a PC with a 3.16 GHz CPU and 3.49 GB of RAM. The initial population is set to 257 and the algorithm is terminated when 10000 iterations have been performed without improving the population.

TABLE II
DESIGN POINTS WITH MINIMUM COST (WITHOUT ANY TIME THRESHOLD)

| Individual # | Total cost | Total time | Possible Assignment of subcontractors to subprojects (24-dimensional vector represents subprojects and the integer values denote the subcontractor assigned to each subproject) |
|--------------|------------|------------|---|
| 1 | 3575.0 | 1418.0 | [1, 5, 5 , 3, 4, 1 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 5, 3 , 3, 1, 2 , 3] |
| 2 | 3575.0 | 1445.0 | [1, 5, 5 , 3, 4, 1 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 1, 5 , 3, 2 , 1, 2 , 3] |
| 3 | 3575.0 | 1485.0 | [1, 5, 5 , 3, 4, 1 , 5, 5, 3 , 1, 2, 4 , 1, 2, 1 , 1, 1, 1, 5 , 3 , 2 , 1, 2 , 3] |
| 4 | 3575.0 | 1458.0 | [1, 5, 5 , 3, 4, 1 , 5, 5, 3 , 1, 2, 4 , 1, 2, 1 , 1, 1, 1, 5, 3 , 3, 1, 2 , 3] |

Bold figures in each row show the longest path (critical path) with respect to completion time of the subprojects

TABLE III
20 DESIGN POINTS THAT FULFILL THE MINIMUM TIME REQUIREMENT = 1192

| Individual # | Total cost | Time Threshold | Possible Assignment of subcontractors to subprojects (24-dimensional vector represents subprojects and the integer values denote the subcontractor assigned to each subproject) |
|--------------|------------|----------------|---|
| 1 | 4470.0 | 1192 | [4 , 3, 3, 5 , 4, 2, 5, 4, 2 , 4, 2 , 5, 1 , 2, 5, 4, 1 , 1, 5, 2 , 1, 1, 5 , 3] |
| 2 | 4471.0 | 1192 | [4 , 3, 3, 5 , 4, 2, 5, 2, 2 , 4, 2 , 5, 1 , 2, 5, 4, 1 , 1, 5, 2 , 1, 1, 5 , 3] |
| 3 | 4479.0 | 1192 | [4 , 3 , 3, 5, 4, 2 , 5, 4, 2, 4 , 2, 5 , 1, 2, 5, 4 , 1, 1, 5 , 5, 1 , 1, 5, 3] |
| 4 | 4480.0 | 1192 | [4, 3 , 3, 5, 4, 2 , 5, 2, 2, 4 , 2, 5 , 1, 2, 5, 4 , 1, 1, 5 , 5, 1 , 1, 5, 3] |
| 5 | 4497.0 | 1192 | [4 , 3, 3, 5 , 4, 2, 3, 4, 2 , 4, 2 , 5, 1 , 2, 5, 4, 1 , 1, 5, 2 , 1, 1, 5 , 3] |
| 6 | 4498.0 | 1192 | [4 , 3, 3, 5 , 4, 2, 3, 2, 2 , 4, 2 , 5, 1 , 2, 5, 4, 1 , 1, 5, 2 , 1, 1, 5 , 3] |
| 7 | 4501.0 | 1192 | [4 , 3, 3, 5 , 4, 2, 5, 4, 2 , 4, 2 , 5, 1 , 2, 2, 4, 1 , 1, 5, 2 , 1, 1, 5 , 3] |
| 8 | 4502.0 | 1192 | [4 , 3, 3, 5 , 4, 2, 5, 2, 2 , 4, 2 , 5, 1 , 2, 2, 4, 1 , 1, 5, 2 , 1, 1, 5 , 3] |
| 9 | 4506.0 | 1192 | [4, 3 , 3, 5, 4, 2 , 3, 4, 2, 4 , 2, 5 , 1, 2, 5, 4 , 1, 1, 5 , 5, 1 , 1, 5, 3] |
| 10 | 4506.0 | 1192 | [4 , 3, 3, 5 , 4, 2, 4, 4, 2 , 4, 2 , 5, 1 , 2, 5, 4, 1 , 1, 5, 2 , 1, 1, 5 , 3] |
| 11 | 4507.0 | 1192 | [4 , 3, 3, 5 , 4, 2, 4, 2, 2 , 4, 2 , 5, 1 , 2, 5, 4, 1 , 1, 5, 2 , 1, 1, 5 , 3] |
| 12 | 4507.0 | 1192 | [4, 3 , 3, 5, 4, 2 , 3, 2, 2, 4 , 2, 5 , 1, 2, 5, 4 , 1, 1, 5 , 5, 1 , 1, 5, 3] |
| 13 | 4510.0 | 1192 | [4, 3 , 3, 5, 4, 2 , 5, 4, 2, 4 , 2, 5 , 1, 2, 2, 4 , 1, 1, 5 , 5, 1 , 1, 5, 3] |
| 14 | 4511.0 | 1192 | [4, 3 , 3, 5, 4, 2 , 5, 2, 2, 4 , 2, 5 , 1, 2, 2, 4 , 1, 1, 5 , 5, 1 , 1, 5, 3] |
| 15 | 4512.0 | 1192 | [4 , 3, 3, 5 , 4, 2, 5, 4, 2 , 4, 2 , 5, 1 , 1, 5, 4, 1 , 1, 5, 2 , 1, 1, 5 , 3] |
| 16 | 4513.0 | 1192 | [4 , 3, 3, 5 , 4, 2, 5, 2, 2 , 4, 2 , 5, 1 , 1, 5, 4, 1 , 1, 5, 2 , 1, 1, 5 , 3] |
| 17 | 4515.0 | 1192 | [4, 3 , 3, 5, 4, 2 , 4, 4, 2, 4 , 2, 5 , 1, 2, 5, 4 , 1, 1, 5 , 5, 1 , 1, 5, 3] |
| 18 | 4516.0 | 1192 | [4, 3 , 3, 5, 4, 2 , 4, 2, 2, 4 , 2, 5 , 1, 2, 5, 4 , 1, 1, 5 , 5, 1 , 1, 5, 3] |
| 19 | 4517.0 | 1192 | [4, 3 , 3, 5, 3, 2 , 5, 5, 2, 4 , 2, 5 , 1, 2, 5, 4 , 1, 1, 5 , 5, 1 , 1, 5, 3] |
| 20 | 4521.0 | 1192 | [4, 3 , 3, 5, 4, 2 , 5, 4, 2, 4 , 2, 5 , 1, 1, 5, 4 , 1, 1, 5 , 5, 1 , 1, 5, 3] |

Bold figures in each row show the longest path (critical path) with respect to completion time of the subprojects

TABLE IV
20 DESIGN POINTS THAT FULFILL REQUIRED TIME THRESHOLD = 1380

| Individual # | Total cost | Total time | Time Threshold | Possible Assignment of subcontractors to subprojects (24-dimensional vector represents subprojects and the integer values denote the subcontractor assigned to each subproject) |
|--------------|------------|------------|----------------|---|
| 1 | 3606.0 | 1373.0 | 1380 | [3, 5, 5 , 3, 4, 2 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 1, 5, 3 , 3, 1, 2 , 3] |
| 2 | 3609.0 | 1368.0 | 1380 | [1, 5, 5, 3 , 4, 2, 5, 5, 3 , 1, 2, 4 , 1, 2, 5 , 1, 1, 1, 5, 3 , 3, 1, 2 , 3] |
| 3 | 3613.0 | 1373.0 | 1380 | [3, 3, 5 , 3, 4, 2 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 1, 5, 3 , 3, 1, 2 , 3] |
| 4 | 3616.0 | 1373.0 | 1380 | [5, 5, 5 , 3, 4, 2 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 1, 5, 3 , 3, 1, 2 , 3] |
| 5 | 3616.0 | 1368.0 | 1380 | [1, 3, 5, 3 , 4, 2, 5, 5, 3 , 1, 2, 4 , 1, 2, 5 , 1, 1, 1, 5, 3 , 3, 1, 2 , 3] |
| 6 | 3617.0 | 1373.0 | 1380 | [3, 5, 5 , 3, 4, 2 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 3, 5, 3 , 3, 1, 2 , 3] |
| 7 | 3620.0 | 1379.0 | 1380 | [1, 5, 5, 3 , 4, 2, 5, 5, 3 , 1, 2, 4 , 1, 2, 5 , 1, 1, 3, 5, 3 , 3, 1, 2 , 3] |
| 8 | 3623.0 | 1373.0 | 1380 | [5, 3, 5 , 3, 4, 2 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 1, 5, 3 , 3, 1, 2 , 3] |
| 9 | 3624.0 | 1380.0 | 1380 | [3, 5, 5, 3, 4, 2 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 1, 5, 1 , 3, 1, 2 , 3] |
| 10 | 3624.0 | 1373.0 | 1380 | [4, 5, 5 , 3, 4, 2 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 1, 5, 3 , 3, 1, 2 , 3] |
| 11 | 3624.0 | 1373.0 | 1380 | [3, 3, 5 , 3, 4, 2 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 3, 5, 3 , 3, 1, 2 , 3] |
| 12 | 3624.0 | 1373.0 | 1380 | [3, 5, 5 , 3, 4, 2 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 1, 5, 3 , 3, 4, 2 , 3] |
| 13 | 3624.0 | 1373.0 | 1380 | [3, 5, 5, 3, 4, 2 , 5, 5, 3 , 1, 2, 4 , 3, 2, 3 , 1, 1, 1, 5, 3 , 3, 1, 2 , 3] |
| 14 | 3627.0 | 1375.0 | 1380 | [1, 5, 5, 3 , 4, 2, 5, 5, 3 , 1, 2, 4 , 1, 2, 5 , 1, 1, 1, 5, 1 , 3, 1, 2 , 3] |
| 15 | 3627.0 | 1379.0 | 1380 | [1, 3, 5, 3 , 4, 2, 5, 5, 3 , 1, 2, 4 , 1, 2, 5 , 1, 1, 3, 5, 3 , 3, 1, 2 , 3] |
| 16 | 3627.0 | 1373.0 | 1380 | [5, 5, 5 , 3, 4, 2 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 3, 5, 3 , 3, 1, 2 , 3] |
| 17 | 3627.0 | 1368.0 | 1380 | [1, 5, 5, 3 , 4, 2, 5, 5, 3 , 1, 2, 4 , 1, 2, 5 , 1, 1, 1, 5, 3 , 3, 4, 2 , 3] |
| 18 | 3627.0 | 1374.0 | 1380 | [3, 5, 5 , 3, 4, 2 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 1, 5, 3 , 3, 1, 2 , 4] |
| 19 | 3628.0 | 1373.0 | 1380 | [3, 5, 5 , 3, 4, 2 , 1, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 1, 5, 3 , 3, 1, 2 , 3] |
| 20 | 3631.0 | 1380.0 | 1380 | [3, 3, 5 , 3, 4, 2 , 5, 5, 3 , 1, 2, 4 , 1, 2, 3 , 1, 1, 1, 5, 1 , 3, 1, 2 , 3] |

Bold figures in each row show the longest path (critical path) with respect to completion time of the subprojects

V. DISCUSSION

In the outsourcing problem discussed in this paper, the manager may be interested to find different assignments of subprojects to subcontractors that satisfy a given time and cost constraint. For instance, assume that it is required that the cost of the project is less than C_u (upper bound on cost) and the time needed to complete the project is less than $T_{threshold}$. These requirements correspond to the region of the solution space shown in Figure 8. Our GA based approach provides several feasible assignments of subprojects to subcontractors and thus the decision makers have the possibility of selecting the most suitable one.

In this example, we assumed that there are m subprojects and n potential subcontractors for each of them. However, a similar problem may arise if there is a single subcontractor for each subproject with agreed upon time and cost, but there is a risk that the completion time is delayed. For instance, assume that company X and subcontractor Y have made the following agreement in their contract: If Y does not complete his work on time, then a penalty of D dollars per day must be paid to X . This will lead to potentially n different costs and completion times for each subproject, where n is the maximum number of days a subcontractor may delay the subproject.

Our approach can also be easily applied to other cases than the outsourcing, where one is interested in finding the subset of the input space (or design points) that is mapped onto a specific region of the solution space, similar to the one shown in Figure 8.

Our GA starts with a randomly selected initial population and then by cross over and mutation operators tries to identify those candidates that fulfill the objective function given in section II. Thus, it is possible that some of the feasible assignments are not detected. This is certainly a limitation of using GA comparing to exhaustive search methods. On the other hand, it is impractical to use exhaustive search when the input space is very large, e.g., in big projects with hundreds of subprojects.

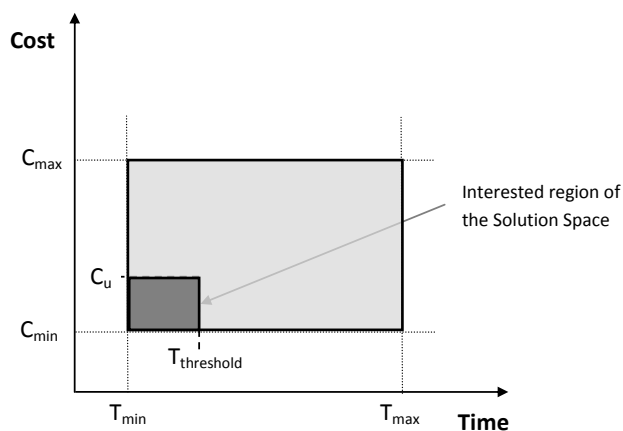


Fig. 8. Interested Region of Solution Space (dark rectangle)

VI. SUMMARY AND CONCLUSION

In many practical situations, the decision makers may be interested to understand the whole solution space, including possible outliers. The Data Farming community has put a lot of efforts in developing methods and techniques for achieving this goal and has used some of the techniques and methods developed in experiment design, such as the Nearly Orthogonal and space-filling Latin Hypercubes (NOLH). In this work, we looked at an outsourcing example but from a different angle. We considered the case where the decision makers are interested in a specific region of the solution space and investigated how to identify those assignments that would be mapped onto this region. We developed a genetic algorithm that randomly selects an initial set of design points and then by cross over and mutation operators tries to identify those candidates that are mapped into the region of interest. We provided a few numerical examples and showed how our GA finds the design points that are mapped into the region of interest. The GA can be used to investigate any part of the solution space, simply by redefining the region of interest.

ACKNOWLEDGMENT

We would like to thank Christian Schulte for his comments and suggestions.

REFERENCES

- [1] G. E. Horne and T. E. Meyer, "Data farming: discovering surprise," in *Proceedings of the 36th conference on Winter simulation*, ser. WSC '04. Winter Simulation Conference, 2004, pp. 807–813.
- [2] J. P. C. Kleijnen, S. M. Sanchez, T. W. Lucas, and T. M. Cioppa, "A users guide to the brave new world of designing simulation experiments," *INFORMS Journal on Computing*, vol. 17, no. 3, pp. 263–289, 2005.
- [3] S. M. Sanchez and H. Wan, "Better than a petaflop: The power of efficient experimental design," in *Proceedings of the 2009 Winter Simulation Conference*, A. Dunkin, R. G. Ingalls, E. Yücesan, M. D. Rossetti, R. Hill, and B. Johansson, Eds., Austin, TX, USA, 2009, pp. 60–74.
- [4] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.
- [5] A. S. Hernandez, "Breaking barriers to design dimensions in nearly orthogonal latin hypercubes," Ph.D. dissertation, Naval Postgraduate School, Monterey, CA, U.S.A., 2008.
- [6] A. K.-E. Joshua, "Extending orthogonal and nearly orthogonal latin hypercube designs for computer simulation and experimentation," Master's thesis, Naval Postgraduate School, Monterey, CA, U.S.A., 2006.
- [7] K. Q. Ye, "Orthogonal column latin hypercubes and their application in computer experiments," *Journal of the American Statistical Association*, vol. 93, no. 444, pp. 1430–1439, 1998.
- [8] T. M. Cioppa, "Efficient nearly orthogonal and space-filling experimental designs for high-dimensional complex models," Ph.D. dissertation, Naval Postgraduate School, Monterey, CA, U.S.A., 2002.
- [9] T. M. Cioppa and T. W. Lucas, "Efficient nearly orthogonal and space-filling latin hypercubes," *Technometrics*, vol. 49, no. 1, pp. 45–55, 2007.
- [10] D. M. Steinberg and D. K. J. Lin, "Orthogonal column latin hypercubes and their application in computer experiments," *Biometrika*, vol. 93, pp. 279–288, 2006.
- [11] M. Mitchell, *Introduction to genetic algorithms*. Cambridge, Massachusetts: MIT Press, 1999.
- [12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Massachusetts: Addison Wesley, 1989.

- [13] J. H. Holland, *Adaptation in natural and artificial systems*. Cambridge, MA, USA: MIT Press, 1992.
- [14] H. Aydt, S. J. Turner, W. Cai, M. Y. H. Low, Y.-S. Ong, and R. Ayani, "Towards an evolutionary computing modeling language," *IEEE Transactions on Evolutionary Computation*, vol. 15, pp. 230–247, April 2011.
- [15] P. C. Chu and J. E. Beasley, "A genetic algorithm for the generalised assignment problem," *Computers and Operations Research*, vol. 24, pp. 17–23, January 1997.
- [16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.
- [17] S. Dasgupta, C. Papadimitriou, and U. Vazirani, *Algorithms*. McGraw-Hill, 2006.
- [18] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991, pp. 69–93.