Interactive Requirements Prioritization using a GA: *Complete Results*

Paolo Tonella, Angelo Susi*, Francis Palma

Software Engineering Research Unit

Fondazione Bruno Kessler

Via Sommarive 18, I-38123, Trento-Povo, Italy

Abstract

The order in which requirements are implemented affects the delivery of value to the end-user, but it also depends on technical constraints and resource availability. The outcome of requirements prioritization is a total ordering of requirements that best accommodates the various kinds of constraints and priorities. During requirements prioritization, some decisions on the relative importance of requirements or the feasibility of a given implementation order must necessarily resort to a human (e.g., the requirements analyst), possessing the involved knowledge.

This report, describes the results obtained by the application of an Interactive Genetic Algorithm (IGA) that includes incremental knowledge acquisition and combines it with the existing constraints, to a real project. We also assess the performance of the proposed algorithm against the state of the art, interactive prioritization technique Incomplete Analytic Hierarchy Process (IAHP).

1. Experimental setting and results

This report describes the results of the application of IGA algorithm to prioritize the requirements for a real software system, as part of the project ACube (Ambient Aware Assistance) [1]. ACube is a large research project funded by the local government of the Autonomous Province of Trento, in Italy, aiming at designing a highly technological smart environment to be deployed in nursing homes

*Corresponding author: Angelo Susi, phone: +39 0461 314344, fax: +39 0461 302040

Email addresses: tonella@fbk.eu (Paolo Tonella), susi@fbk.eu (Angelo Susi), fpalma@fbk.eu (Francis Palma)

to support medical and assistance staff. In such context, an activity of paramount importance has been the analysis of the system requirements, to obtain the best trade off between costs and quality improvement of services in specialized centers for people with severe motor or cognitive impairments. From the technical point of view, the project envisages a network of sensors distributed in the environment or embedded in the end-users' clothes. This technology should allow monitoring the nursing home guests unobtrusively, that is, without influencing their usual daily life activities. By means of advanced automatic reasoning algorithms, the data acquired through the sensor network are going to be used to promptly recognize emergency situations and to prevent possible dangers or threats for the guests themselves. The ACube project consortium has a multidisciplinary nature, involving software engineers, sociologists and requirements analysts, and is characterized by the presence of professionals representing end-users directly engaged in design activities.

As a product of the end-user requirements analysis phase, 60 end-user requirements (49 technical requirements[1]) and three macro-scenarios have been identified. Specifically, the three macro scenarios are: (i) "localization and tracking to detect falls of patients", (ii) "localization and tracking to detect patients escaping from the nursing home", (iii) "identification of dangerous behavior of patients"; plus (iv) a comprehensive scenario that involves the simultaneous presence of the previous three scenarios.

Out of these macro-scenarios, detailed scenarios have been analyzed together with the 49 technical requirements. Examples of such technical requirements are:

> *"TR16: The system identifies the distance between the patient and the nearest healthcare operator"*

or

> *"TR31: The system infers the kind of event based on the available information"*

Table 1 summarizes the number of technical requirements for each macro-scenario. Together with the set of technical requirements, two sets of technical constraints have been collected during requirements elicitation: *Priority* and *Dependency*, representing respectively the priorities among requirements and their dependencies. In particular, the *Priority* constraint has been built by the software

[1]We consider only functional requirements.

Id	Macro-scenario	Number of requirements
FALL	Monitoring falls	26
ESC	Monitoring escapes	23
MON	Monitoring dangerous behavior	21
ALL	The three scenarios	49

Table 1: The four macro-scenarios and the number of technical requirements associated with them.

architect on the basis of the end-users' needs and it is defined as a function that associates each technical requirement to a number (in the range 1–500), indicating the priority of the technical requirement with respect to the priority of the end-user requirements it is intended to address. The *Dependency* feature is defined on the basis of the dependencies between requirements and is a function that links a requirement to the set of other requirements it depends on.

Finally, for each of the four macro-scenarios, we obtained the *Gold Standard* (GS) prioritization from the software architect of the ACube project. The GS prioritization is the ordering given by the software architect to the requirements when he planned their implementation during the ACube project. We take advantage of the availability of GS in the experimental evaluation of the proposed algorithm, in that we are able to compare the final ordering produced by the algorithm with the one defined by the software architect.

1.1. Research questions

The experiments we conducted aim at answering the following research questions:

RQ1 (Convergence) *Can we observe convergence with respect to the finally elicited fitness function?*

Since the fitness function is constructed incrementally during the interactive elicitation process, convergence is not obvious. In fact, initially IGA optimizes the ordering so as to minimize the disagreement with the available precedence graphs (*Priority* and *Dependency* in our case study). Then, constraints are added by the user (in our case the ideal user simulated by sampling the elicited pairs from the Gold Standard) and a new precedence graph (*eliOrd*) appears. Hence, the target of optimization is slightly and gradually changed. The question is whether the overall optimization process converges, once we consider (a-posteriori) all prece-

dence graphs, including the elicited one in its final form. We answer this research question by measuring the final fitness function values (i.e., disagreement with all final precedence graphs) over generations after the evolutionary process is over (hence, *eliOrd* has been gathered completely). It should be noticed that the final fitness function values are not available during the evolutionary optimization process, when they are approximated as the disagreement with the initial precedence graphs and the partially constructed *eliOrd*.

RQ2 (Comparison) *Does IGA produce improved prioritizations compared to IAHP?*

In our previous work [2] we showed that IGA outperforms non-interactive requirement ordering. In this paper, we compare IGA with another interactive requirement prioritization technique, IAHP [3]. IAHP (Incomplete AHP) is a variant of AHP [4] that can deal with incomplete information (pairwise comparisons) from the user. It was designed to support scalability of AHP to requirements whose size make AHP prohibitively expensive (the number of pairs that must be elicited to apply AHP is quadratic with the number of requirements). It represents the state of the art in interactive requirements prioritization. Appendix A provides a brief introduction to the IAHP algorithm.

In this research question we compare the output of IGA with the output of the IAHP algorithm, at equal number of pairs elicited from the user. We vary such number from low to high values, in order to investigate the regions where one approach is superior to the other (i.e., the degree of information incompleteness each approach can accommodate).

To assess the performance of IGA and IAHP we use two main metrics: number of disagreements between the rank and GS and average distance of the position of a requirement in the rank from the position of the same requirement in the GS. The latter metrics is highly correlated with the former, but it has the advantage of being more easily interpretable than disagreement. In fact, disagreement involves a quadratic number of comparisons (each pair of requirements w.r.t. GS order), hence its absolute value is not straightforward to understand and compare. On the contrary, the distance between the position of each requirement in the prioritization produced by our algorithm and the position of the same requirement in the GS gives a direct clue to the number of requirements that are incorrectly positioned before or after the requirement being considered.

RQ3 (Role of weights) *How does the specification of weights affect the performance of IGA?*

The fitness function used by the IGA algorithm may be improved by specifying the relative importance of the various precedence graphs used for prioritization (including the elicited precedence graph).

In this evaluation, we exploited the disagreement measure defined in the general case where two partial orders are compared.

$$dis(ord_1, ord_2) = \{(p, q) \in ord_1^* \mid (q, p) \in ord_2^*\} \quad (1)$$

The disagreement between two (partial or total) orders ord_1, ord_2, defined upon the same set of elements R, is the set of pairs in the transitive closure[2] of the first order, ord_1^*, that appear reversed in the second order closure ord_2^*. A measure of disagreement is given by the size of the set $dis(ord_1, ord_2)$, under the assumption that all pairs are weighted the same. However, sometimes it is useful to give different weights to different precedence graphs, or even to specific edges of a given precedence graph.

This research question deals with the role of such weights. We want to understand whether the specification of weights can speed up the convergence to the final solution in a significant way. To this aim, we consider some alternative settings for the weights to be given to *Prio*, *Dep* and *Eli*. We discussed the considered settings with the system architect and the requirements analysts of Acube, so as to investigate those configurations which better match the relative importance of the available information, as perceived by the people who actually worked on the project.

RQ4 (Robustness) *Is IGA more robust than IAHP with respect to errors committed by the user during the elicitation of pairwise comparisons?*

In order to test the robustness of the IGA in comparison with IAHP at increasing user error rates, we simulate such errors by means of a simple stochastic model. We fix the probability of committing an elicitation error (p_e). Then, during the execution of the IGA and IAHP algorithms, whenever a pairwise comparison is elicited from the user, we generate a response in agreement with the GS with probability $1 - p_e$ and we generate an error (i.e., a response in disagreement with the GS) with probability p_e. We varied the probability of user error p_e from 5% to

[2]The transitive closure is defined as follows: $(p, q) \in ord^*$ *iff* $(p, q) \in ord$ or $\exists r | (p, r) \in ord \wedge (r, q) \in ord^*$.

20%.

1.2. Experiments performed

We executed several experiments to answer the research questions listed above. In this section we summarize the experimental settings used in each experiment.

1.2.1. Experimental design

The factors to be tested through our experimentation include: (a) *Applied Algorithm/Method*, for the experimental design we used our approach i.e. IGA along with Incomplete-AHP (IAHP). We compare resulting disagreement/distance of the final ordering produced as the outcome of the experimented approaches w.r.t. GS; (b) *User Model*, in our experiments we consider both the case where there is an ideal user (e.g. user makes no error) and a user that sometimes makes wrong decisions; to simulate such a user we resort to the GS ordering of pairs (or its opposite, if the simulated user is supopsed to make an error); and, (c) *Available Knowledge*, possibility to take advantage of different sources of knowledge (i.e. user knowledge or domain knowledge, coming from the requirements analysis phase) and to give them different weights.

1.2.2. Experimental Preparation: The levels of the factors

(a) *Applied Algorithm/Method*: The algorithm or method that we apply is the main factor (i.e. we expect the use of different algorithms or methods will produce different levels of disagreement/distance; see Table 2).

Factor	**Levels**
Algorithm	IGA (Interactive, Domain Info, User Knowledge through pairwise comparison)
	IAHP (Interactive, User Knowledge through pairwise comparison)

Table 2: Levels for the factors used in IGA & IAHP

(b) *User Model*: In our experiment the user is an artificial user agent that replies to the requests of pairwise comparisons automatically, either in agreement with the GS (with probability $1 - p_e$), or in disagreement with it (with probability p_e). We have conducted experiments with all four macro scenarios (see Table 1) with a different number of pairs elicited from the user and with variations in user error rates. More specifically, for all macro scenarios we experimented with 25,

50 & 100 elicited pairs and error rates 0% (no errors), 5% 10% and 20%. Table 3 reflects the user and error model we used in our experiments (numbers remain same for all macro scenarios i.e. ALL, FALL, ESC & MON).

Factor	**Levels**		
User model	***Tot. Eli. Pairs***	***Err. Rate***	***Wrong Eli. Pairs***
	25	0%	0
		5%	1
		10%	2
		20%	5
	50	0%	0
		5%	2
		10%	5
		20%	11
	100	0%	0
		5%	5
		10%	10
		20%	20

Table 3: Parameters of the user model used in the experiments

In Table 3, the second column represents the total number of elicited pairs we experimented with. The third column represents the user error rates we used and the fourth column represents the calculated number of elicited pairs that are answered wrongly by the artificial user agent at runtime.

Weight model	Prio	Dep	Eli
Base model (BM)	1	1	1
Weight model 1 (WM1)	2	1	4
Weight model 2 (WM2)	2	1	2
Weight model 3 (WM3)	2	1	1

Table 4: Relative weights given to Prio, Dep and Eli constraint graphs

(c) *Weights*: We investigated three weight models that were regarded as interesting variants by the system architect and by the requirements analysts of the Acube project. They are listed in Table 4. They all give higher importance to the end-user's preferences (Prio), compared to the implementation constraints (Dep), which makes sense for a user-centered system such as Acube. The first one, which

gives the highest importance (weight = 4) to the elicited constraints, is indeed the one suggested by the system architect and by most requirements analysts. The other two models, which gradually reduce the importance attributed to the elicited constraints while keeping a 2:1 ratio between Prio and Dep, have been suggested as variants of the first model that are worth to be investigated.

1.2.3. Experimental Measures

To evaluate the outcome (i.e., the prioritized list of requirements) of the experimented methods, we considered the measurement scales reported in Table 5. The average distance AD is computed by considering the difference between the positions of each requirement R_i in two total orders P_1, P_2, and taking the average over the requirements:

$$AD(P_1, P_2) = \frac{1}{n} \sum_{i=1}^{n} |pos(P_1, R_i), pos(P_2, R_i)| \quad (2)$$

Measurement	**Type**	**Description**
Disagreement (DIS)	Count	Total count of mismatched pairs w.r.t. GS
Average Distance (AD)	Position	Position distance of each requirement w.r.t. its position in the GS

Table 5: Different types of measurements used in the experiments with IGA & IAHP

1.2.4. Experimental Settings

The experimental settings used to answer the research questions in the previous section are reported in Table 6. These parameters were used in all the scenarios considered in the experiments (ALL, FALL, ESC, MON). GA parameters are based on values commonly used in the literature and on a few preliminary calibration runs of the algorithm, in accordance with the offline parameter tuning method [5].

Parameters	Values
targetAlgorithm	IGA, IAHP
populationSize (IGA)	50
mutationRate (IGA)	10%
maxElicitedPairs	25, 50, 100
executionTime (for IGA)	600s, 840s & 1080s
userErrorRate	0%, 5%, 10%, 20%
targetMeasure	DIS, AD

Table 6: Different experimental settings used for experimentation with IGA & IAHP

1.2.5. Experimental Executions

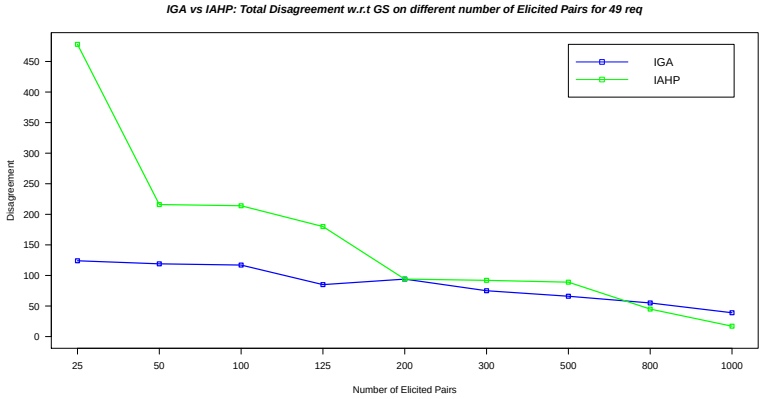
Since IGA is non deterministic, we replicated each experiment a minimum number of 10 to maximum of 20 times with a time-out between 600s and 1080s. We varied the maximum execution-time in accordance with the maximum number of elicited pairs. The more pairs we elicit, the more operational time is assumed to be available. So, we assumed a range of 600s, 840s and 1080s as execution time while eliciting 25, 50 & 100 pairs respectively. The maximum number of elicited pairs is set to 100 based on our previous experience with pairwise comparison experiments involving humans. Above this threshold, results become unreliable because of fatigation, distraction and boredom. Our experience is corroborated by available cognitive psychology studies on user attention and performance [6]. For IAHP, we also performed experiments on 10 random initial orderings to observe its consistency w.r.t. the initialization and we terminate the prioritization process when we reach *maxElicitedPairs*. For both algorithms, we simulate the artificial user who automatically responds according to the GS both in error-free setting and in erroneous settings. After the optimization process terminates, we computed disagreement and average distance for the returned prioritized order w.r.t. GS. Finally, we produced box-plots for all the results we obtained, to be able to visually compare IGA and IAHP (or the other factors being investigated). We also executed statistical tests (e.g., ANOVA) to check whether the visually observed differences were also statistically significant.

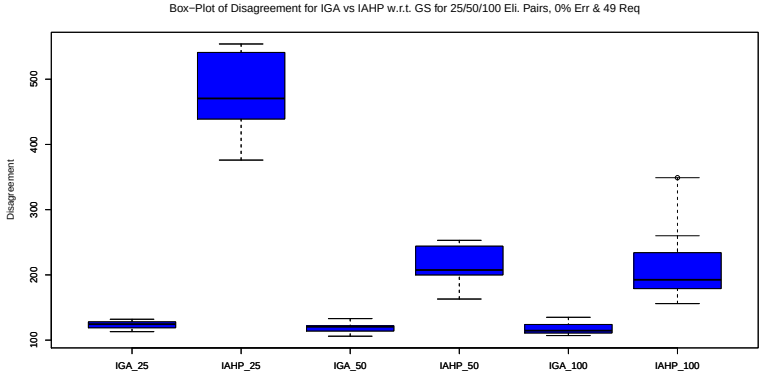
2. Results

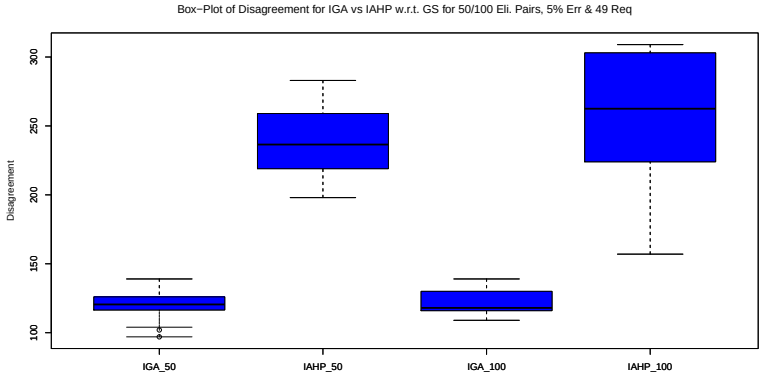
Here we report all the results for all the scenarios considered in our experiment (ALL, MON, FALL and ESC).

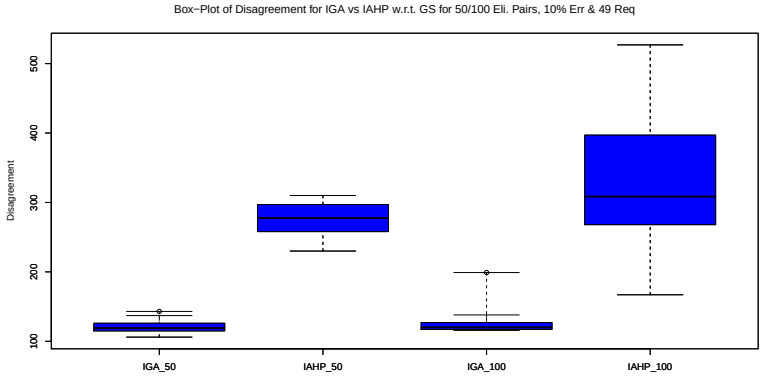
2.1. ALL scenario

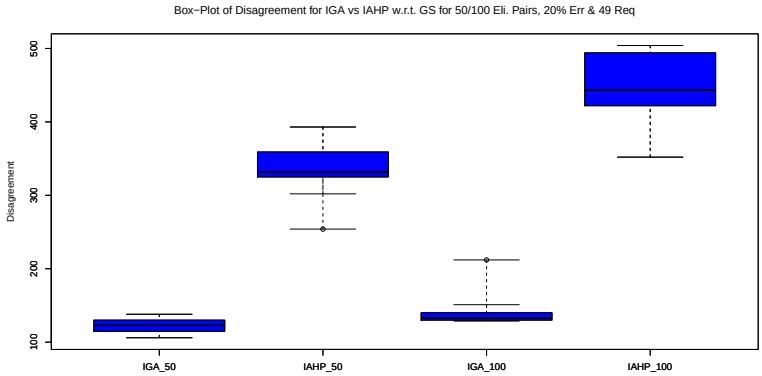
Here in the following the results of the experiments on the ALL scenario.

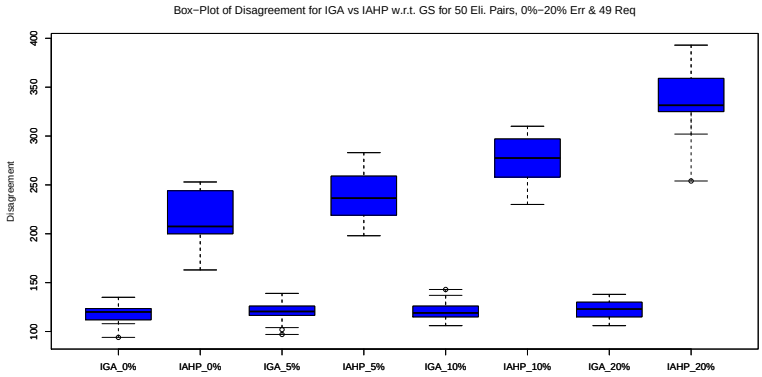


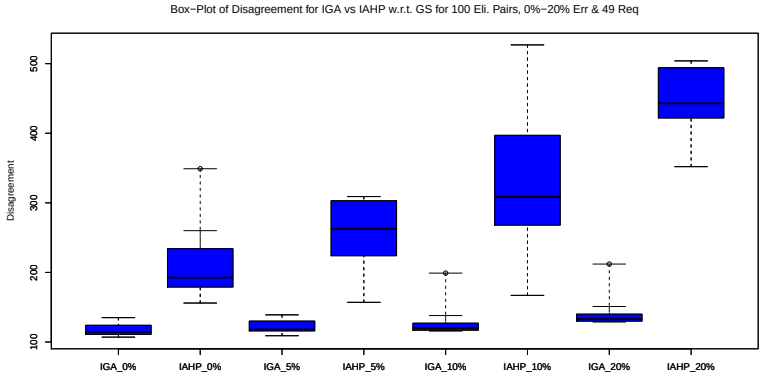


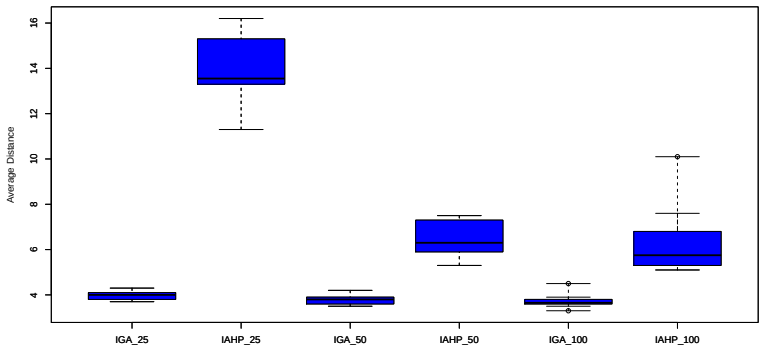


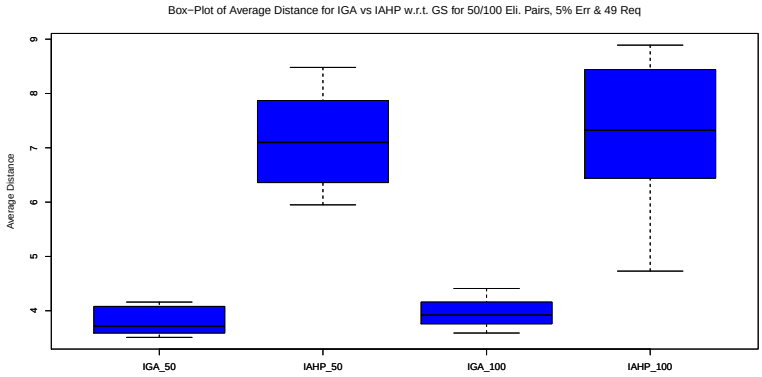

Box−Plot of Disagreement for IGA vs IAHP w.r.t. GS for 50/100 Eli. Pairs, 10% Err & 49 Req
Disagreement
100
200
300
400
500
IGA_50
IAHP_50
IGA_100
IAHP_100

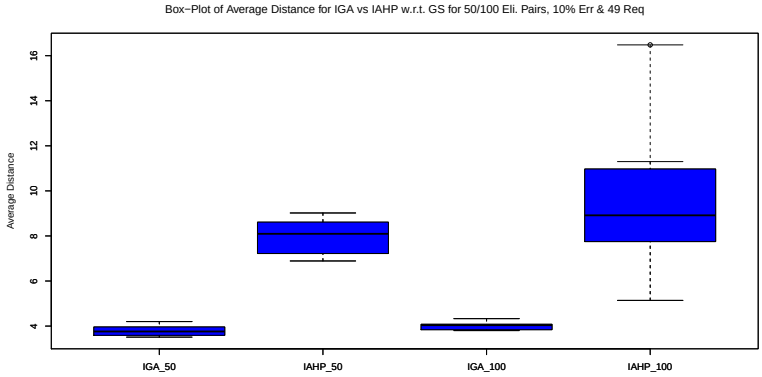

Box−Plot of Disagreement for IGA vs IAHP w.r.t. GS for 50/100 Eli. Pairs, 20% Err & 49 Req
Disagreement
100
200
300
400
500
IGA_50
IAHP_50
IGA_100
IAHP_100

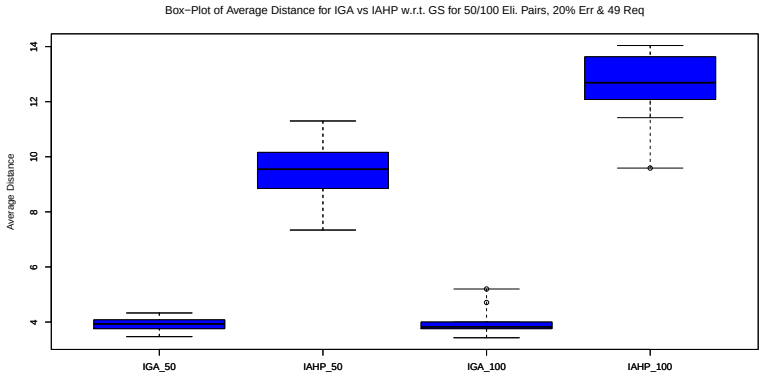

Box−Plot of Disagreement for IGA vs IAHP w.r.t. GS for 50 Eli. Pairs, 0%−20% Err & 49 Req
Disagreement
100
150
200
250
300
350
400
IGA_0%
IAHP_0%
IGA_5%
IAHP_5%
IGA_10%
IAHP_10%
IGA_20%
IAHP_20%

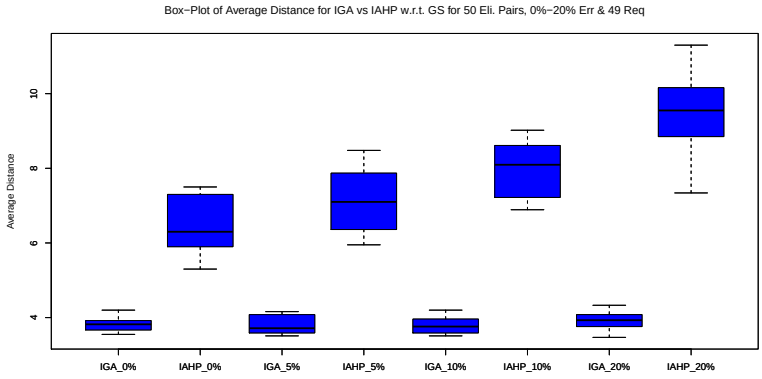

Box−Plot of Disagreement for IGA vs IAHP w.r.t. GS for 100 Eli. Pairs, 0%−20% Err & 49 Req
Disagreement
100
200
300
400
500
IGA_0%
IAHP_0%
IGA_5%
IAHP_5%
IGA_10%
IAHP_10%
IGA_20%
IAHP_20%

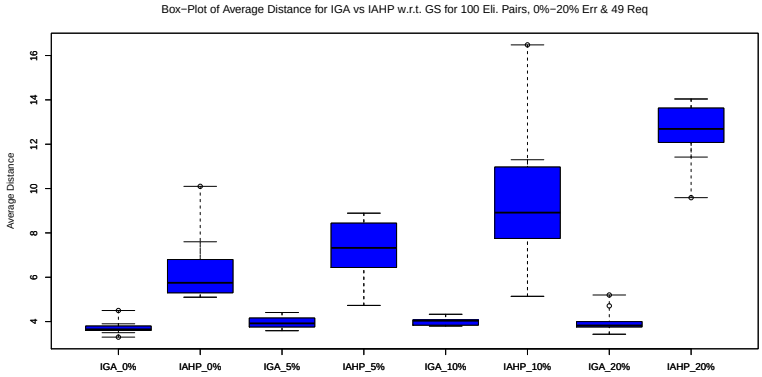

Box−Plot of Average Distance for IGA vs IAHP w.r.t. GS for 25/50/100 Eli. Pairs, 0% Err & 49 Req
Average Distance
4
6
8
10
12
14
16
IGA_25
IAHP_25
IGA_50
IAHP_50
IGA_100
IAHP_100


Box−Plot of Average Distance for IGA vs IAHP w.r.t. GS for 50/100 Eli. Pairs, 5% Err & 49 Req
Average Distance
4
5
6
7
8
9
IGA_50
IAHP_50
IGA_100
IAHP_100


Box−Plot of Average Distance for IGA vs IAHP w.r.t. GS for 50/100 Eli. Pairs, 10% Err & 49 Req
Average Distance
4
6
8
10
12
14
16
IGA_50
IAHP_50
IGA_100
IAHP_100

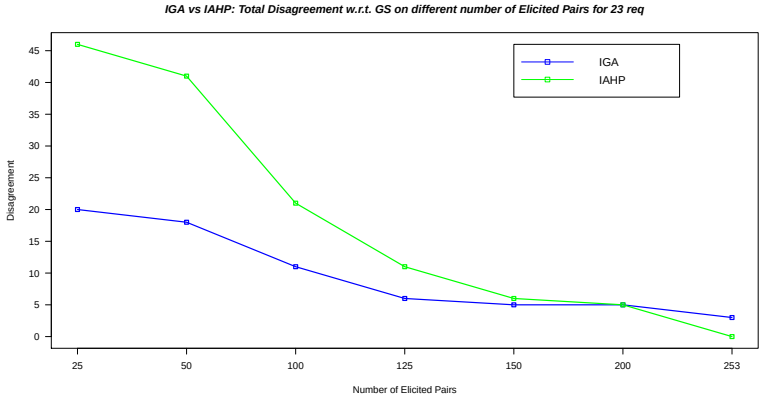

Box−Plot of Average Distance for IGA vs IAHP w.r.t. GS for 50/100 Eli. Pairs, 20% Err & 49 Req
Average Distance
4
6
8
10
12
14
IGA_50
IAHP_50
IGA_100
IAHP_100

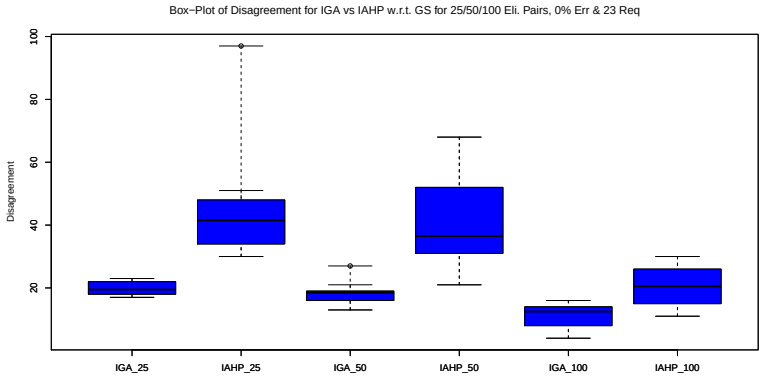

Box−Plot of Average Distance for IGA vs IAHP w.r.t. GS for 50 Eli. Pairs, 0%−20% Err & 49 Req
Average Distance
4
6
8
10
IGA_0%
IAHP_0%
IGA_5%
IAHP_5%
IGA_10%
IAHP_10%
IGA_20%
IAHP_20%

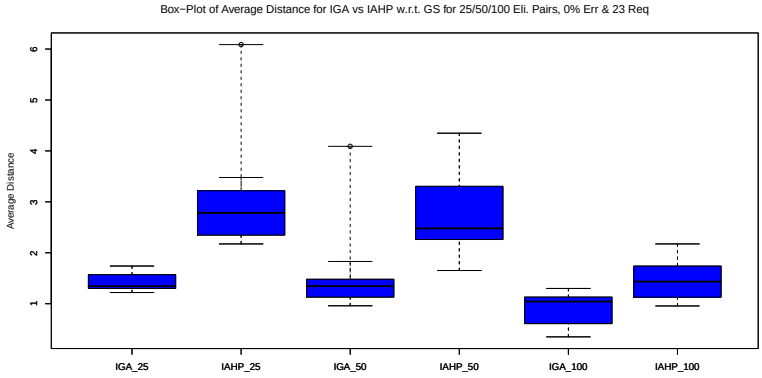

Box−Plot of Average Distance for IGA vs IAHP w.r.t. GS for 100 Eli. Pairs, 0%−20% Err & 49 Req
Average Distance
4
6
8
10
12
14
16
IGA_0%
IAHP_0%
IGA_5%
IAHP_5%
IGA_10%
IAHP_10%
IGA_20%
IAHP_20%

2.2. *ESC scenario*

Here in the following the results of the experiments on the ESC scenario.

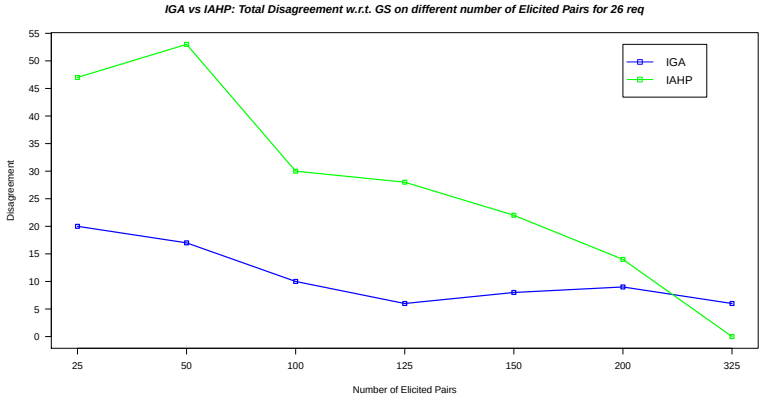


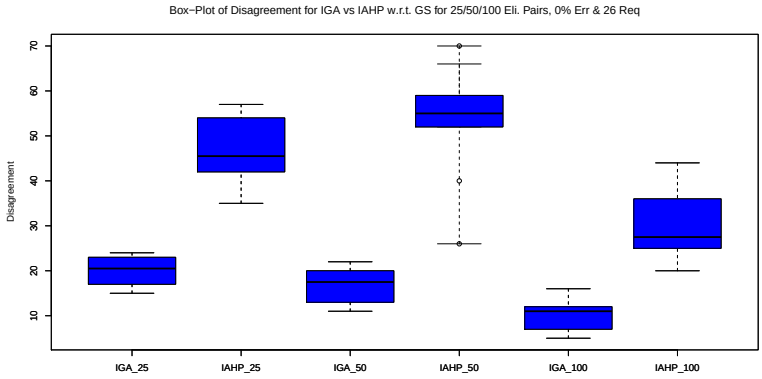


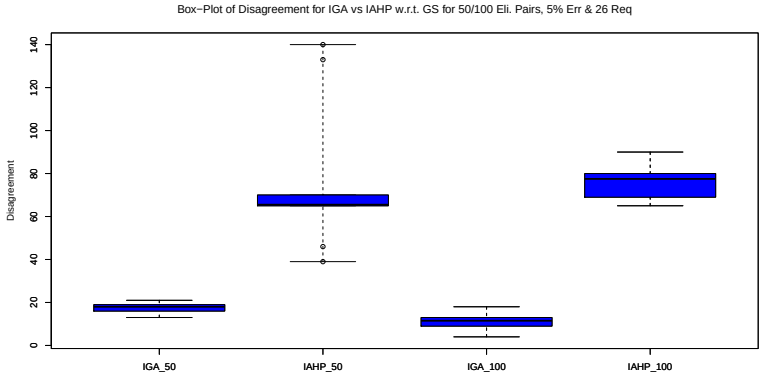


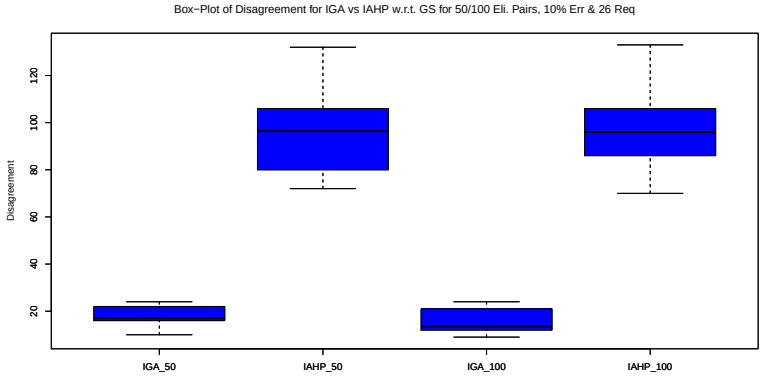
2.3. FAL scenario

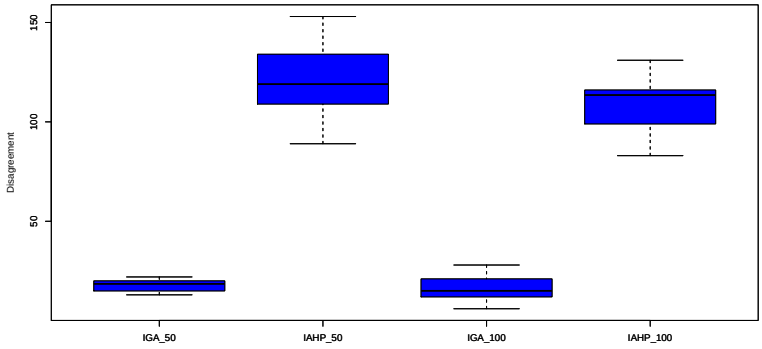
Here in the following the results of the experiments on the FAL scenario.

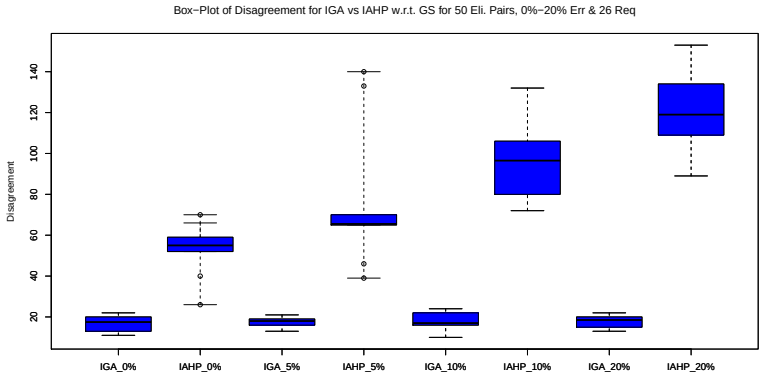


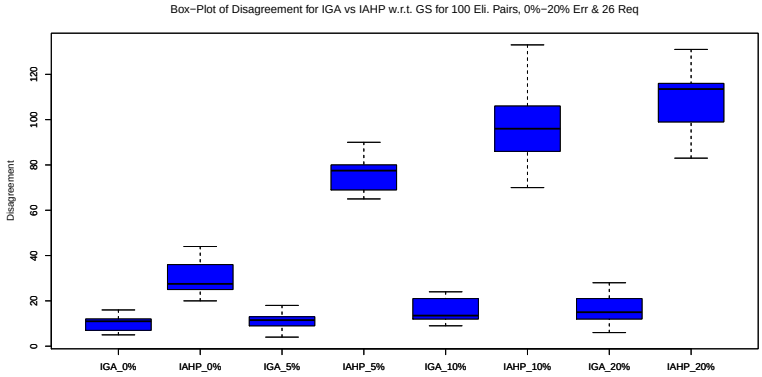


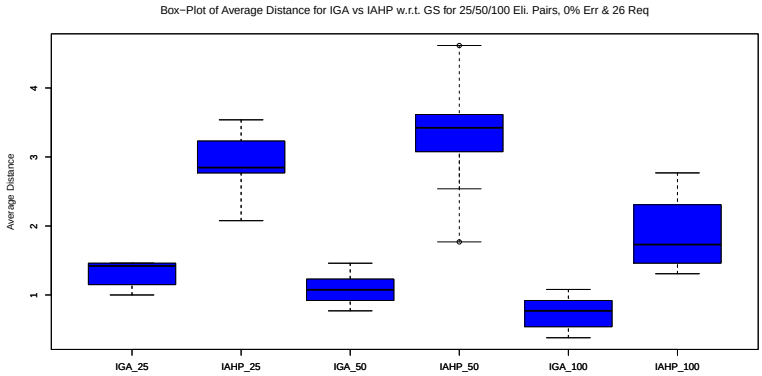


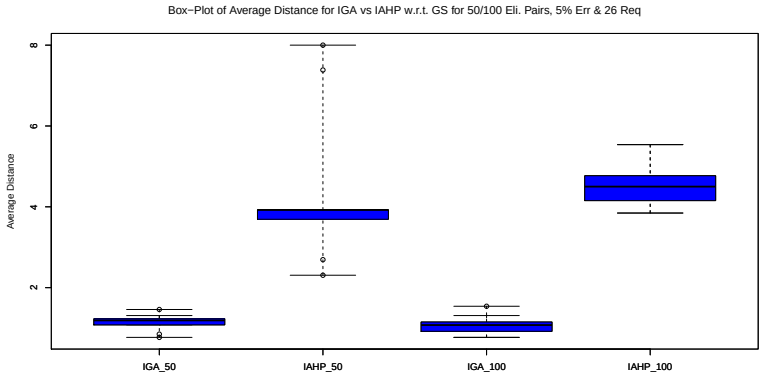

Box−Plot of Disagreement for IGA vs IAHP w.r.t. GS for 50/100 Eli. Pairs, 10% Err & 26 Req
Disagreement
20
40
60
80
100
120
IGA_50
IAHP_50
IGA_100
IAHP_100

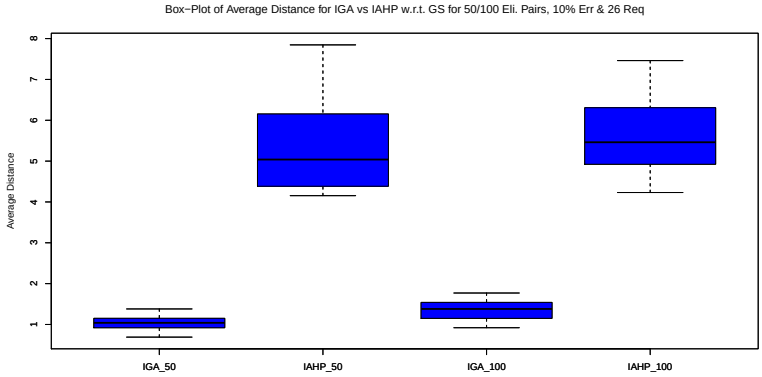

Box−Plot of Disagreement for IGA vs IAHP w.r.t. GS for 50/100 Eli. Pairs, 20% Err & 26 Req
Disagreement
50
100
150
IGA_50
IAHP_50
IGA_100
IAHP_100

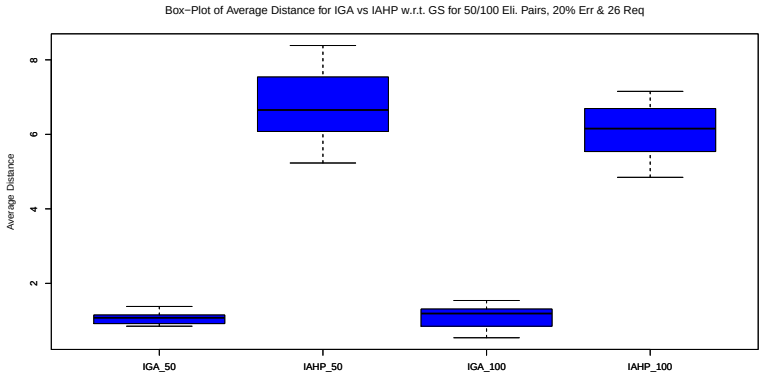

Box−Plot of Disagreement for IGA vs IAHP w.r.t. GS for 50 Eli. Pairs, 0%−20% Err & 26 Req
Disagreement
20
40
60
80
100
120
140
IGA_0%
IAHP_0%
IGA_5%
IAHP_5%
IGA_10%
IAHP_10%
IGA_20%
IAHP_20%

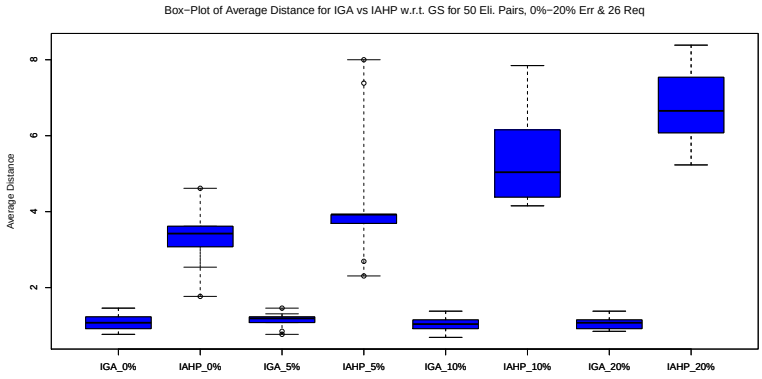

Box−Plot of Disagreement for IGA vs IAHP w.r.t. GS for 100 Eli. Pairs, 0%−20% Err & 26 Req
Disagreement
0
20
40
60
80
100
120
IGA_0%
IAHP_0%
IGA_5%
IAHP_5%
IGA_10%
IAHP_10%
IGA_20%
IAHP_20%

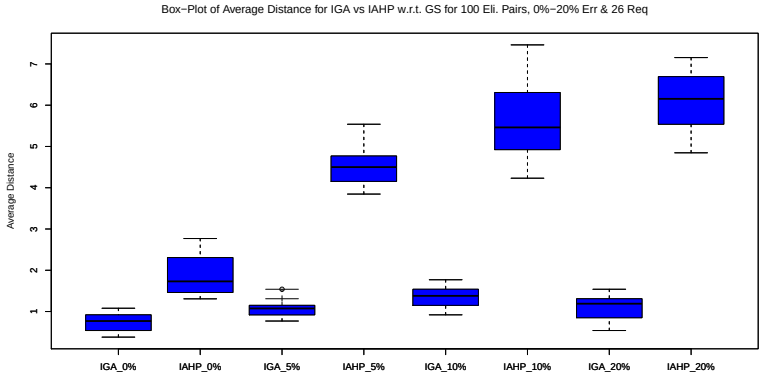

Box−Plot of Average Distance for IGA vs IAHP w.r.t. GS for 25/50/100 Eli. Pairs, 0% Err & 26 Req
Average Distance
1
2
3
4
IGA_25
IAHP_25
IGA_50
IAHP_50
IGA_100
IAHP_100


Box−Plot of Average Distance for IGA vs IAHP w.r.t. GS for 50/100 Eli. Pairs, 5% Err & 26 Req
Average Distance
2
4
6
8
IGA_50
IAHP_50
IGA_100
IAHP_100


Box−Plot of Average Distance for IGA vs IAHP w.r.t. GS for 50/100 Eli. Pairs, 10% Err & 26 Req
Average Distance
1
2
3
4
5
6
7
8
IGA_50
IAHP_50
IGA_100
IAHP_100

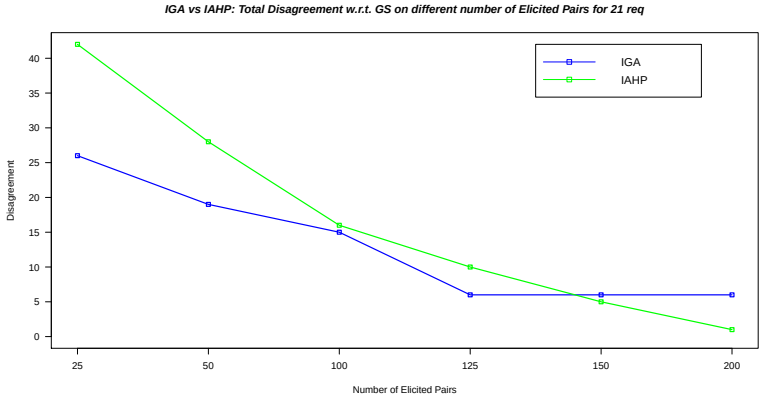

Box−Plot of Average Distance for IGA vs IAHP w.r.t. GS for 50/100 Eli. Pairs, 20% Err & 26 Req
Average Distance
2
4
6
8
IGA_50
IAHP_50
IGA_100
IAHP_100

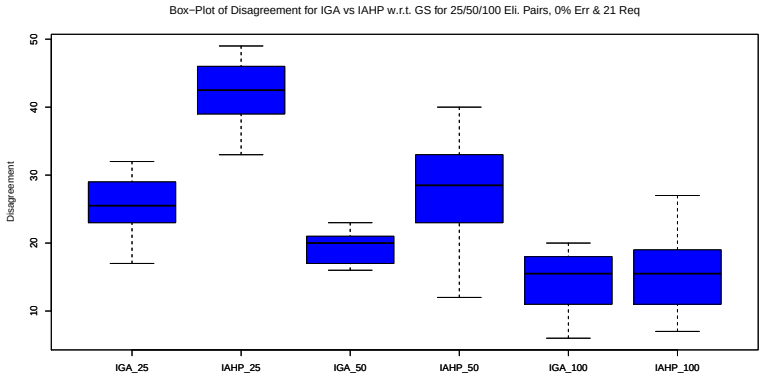

Box−Plot of Average Distance for IGA vs IAHP w.r.t. GS for 50 Eli. Pairs, 0%−20% Err & 26 Req
Average Distance
2
4
6
8
IGA_0%
IAHP_0%
IGA_5%
IAHP_5%
IGA_10%
IAHP_10%
IGA_20%
IAHP_20%

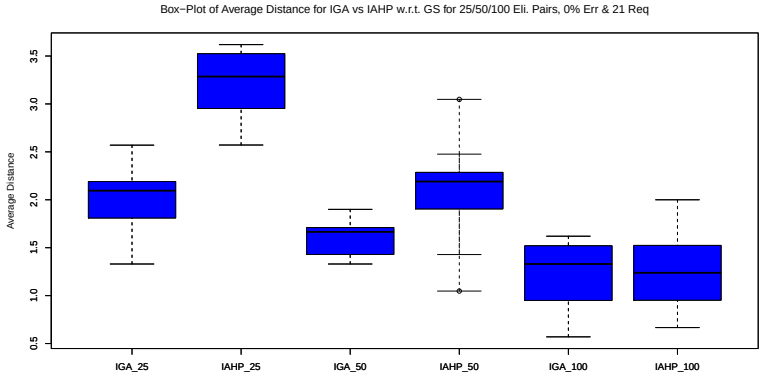

Box−Plot of Average Distance for IGA vs IAHP w.r.t. GS for 100 Eli. Pairs, 0%−20% Err & 26 Req
Average Distance
1
2
3
4
5
6
7
IGA_0%
IAHP_0%
IGA_5%
IAHP_5%
IGA_10%
IAHP_10%
IGA_20%
IAHP_20%

2.4. *MON scenario*

Here in the following the results of the experiments on the MON scenario.







References

[1] R. Andrich, F. Botto, V. Gower, C. Leonardi, O. Mayora, L. Pigini, V. Revolti, L. Sabatucci, A. Susi, M. Zancanaro, ACube: User-Centred and Goal-Oriented techniques, Technical Report, Fondazione Bruno Kessler - IRST, 2010.

[2] P. Tonella, A. Susi, F. Palma, Using Interactive GA for Requirements Prioritization, in: 2nd International Symposium on Search Based Software Engineering, IEEE, pp. 57–66.

[3] P. T. Harker, Incomplete Pairwise Comparisons in the Analytic Hierarchy Process, Math. Modelling 9 (1987) 837 – 848.

[4] T. L. Saaty, L. G. Vargas, Models, Methods, Concepts & Applications of the Analytic Hierarchy Process, Kluwer Academic, 2000.

[5] A. E. Eiben, S. K. Smit, Evolutionary algorithm parameters and methods to tune them, in: E. M. Y. Hamadi, F. Saubion (Eds.), Autonomous Search, Springer, 2011.

[6] D. A. Allport, Attention and performance, in: G. Claxton (Ed.), Cognitive Psychology: New Directions, Routledge & Kegan Paul, London, UK, 1980.

[7] J. Karlsson, K. Ryan, A cost-value approach for prioritizing requirements, IEEE Software 14 (1997) 67–74.

Appendix A. AHP and incomplete AHP

Here we briefly introduce the Analytical Hierarchy Process (AHP) [4] method and its incomplete version defined by Harker [3], that represent the state-of-the-art in interactive requirements prioritization.

Appendix A.1. AHP method

AHP was developed by Thomas Saaty and was first applied to requirement prioritization by Karlsson and Ryan [7]. AHP can be considered one of the reference methods adopting a pairwise comparison strategy, allowing to define the prioritization criteria through a priority assessment of all the possible pairs of requirements.

Given a set of n requirements $R = \{R_1, \ldots, R_i, R_j, \ldots, R_n\}$, the first step in AHP is the construction of an $n \times n$ matrix whose rows and columns represent the candidate requirements. In this step a pairwise comparison iterative process is used in which the decision maker is required to give an integrer value for preference $p_{ij} \in [1 \ldots 9]$, between two requirements, for each possible pairs of requirements. The value represents a qualitative measure of the preference relation, so, given two requirements R_i and R_j if the requirement R_i is "equally important" as requirement R_j with respect to the given criterion, $p_{ij} = 1$ is given, if the requirement R_i is "moderatly more important" than requirement R_j the value $p_{ij} = 3$ is given, if the requirement R_i is "strongly more important" than requirement R_j the value $p_{ij} = 5$ is given, and so on; a set of possible values are presented in Table A.7. The elicited value p_{ij} is inserted in the corresponding cell of the matrix (R_i, R_j), while the cell (R_j, R_i) is filled with the reciprocal of the value $p_{ji} = 1/p_{ij}$. When all the pairs have been evaluated, a ranking is synthesized through the computation of the principal eigenvector of the matrix (i.e., the eigenvector with the highest eigenvalue norm). Each component of the principal eigenvector represents the rank of each requirement.

AHP has the important property of detecting the inconsistencies of the decision maker while eliciting priorities, providing a way to measure the confidence in the prioritization results. On the other hand, this method becomes of difficult application as soon as the number of requirements increases, thus inducing scalability problems. In fact, the user has to elicit $n(n-1)/2$ pair-wise comparisons to compute the solution.

Appendix A.2. IAHP method

To overcome the AHP scalability issues, Harker [3] proposed an incomplete version of AHP (IAHP). This method relies on the key idea of minimizing the

Preference p_{ij}	**Definition**
1	R_i Equally important to R_j
3	R_i Moderatly more important than R_j
5	R_i Strongly more important than R_j
7	R_i Very strongly or demonstrated more important than R_j
9	R_i Extremely more important than R_j
2, 4, 6, 8	For compromise between the above values

Table A.7: A possible fundamental scale in the interval $[1 \dots 9]$ used for AHP

number of pairs elicited from the decision maker, maintaining, at the same time, a good trade-off between the precision of the final solution and the effort of the decision maker. This is done by calculating, at each iteration of the elicitation process, a prediction of the next most promising pair to be asked to the decision maker in order to find a stable and good approximation of the target ranking, early before eliciting all the $n(n-1)/2$ pairs.

In particular, the steps used in IAHP are:

1. The decision maker provides $n-1$ judgments which form a spanning tree. The n nodes of the tree are the requirements. The $n-1$ edges represent the pairwise comparisons elicited from the user. Each edge has an intensity taken from the same scale as the one used by AHP and shown in Table A.7.
2. Using the available pairwise comparisons, the missing comparisons are derived by taking the geometric mean of the intensities, computed on a sample of the paths that connect start and end node (for each pair of nodes not connected by an edge). With these estimated intensities, the weight matrix is complete.
3. The derivatives of the weight matrix are computed with respect to the missing matrix elements and the next question (pairwise comparison to elicit) is selected as the one which maximizes the norm of the derivative (hence having the largest impact on the estimated intensities).
4. If the stopping criterion is met (e.g., maximum number of pairs has been elicited), then the algorithm stops and the final rank is calculated from the principal eigenvector of the matrix (as in AHP), else the selected pairwise comparison is elicited and the algorithm continues with step 2.

Other stopping criteria may be adopted, instead of the maximum number of pairwise comparisons. For instance, if the maximum absolute difference in the

estimated weights from one question to another is below a given threshold, we can assume that convergence has already been reached. Another convergence indicator is that the ordinal ranking given by the principal eigenvector components remains unchanged.