Veridian E-Journal, Science and Technology Silpakorn University
Volume 5 Number 5 September – October 2018 ISSN 2408 - 1248

A Method to Reduce Semantic Redundancy of Question in Bee Algorithm-based Test Assembly Task

Thanuwong Chaksupa*

Abstract

Redundancy of questions in a test-form generated from an automated test assembly is a problem leading to unfairness in an examination. This paper thus proposes a method to enhance automated test assembly using semantic similarity calculation for reducing a probability in assembling similar questions in the same test-form. The semantic similarity is designed to use as an objective function in Bee algorithm which reliably performs combinatorial optimization to manage computation complexity and required constraints in test assembling. The experiments were conducted with 400 questions in information technology domain in comparison of using and not using the semantic similarity. The results showed that applying semantic similarity calculation in a test assembly could greatly reduce a number of co-occurred similar questions for more than 90% in average. In addition, it does not increase computational times, and it can produce test-forms meeting specified constraints such as an equivalently distributed test length.

Keywords: Automated Test Assembly, Semantic Similarity, Redundancy Reduction, Test Design

* Dr.Thanuwong Chaksupa, Department of Computer, Faculty of Liberal Arts and Science, Kasetsart University Khamphaengsaen, faastwc@gmail.com, 084-1568226

Introduction

A test assembly is a method to generate several test-forms from a question pool (Armstrong et al,1992; van der Linden, 2005, Belov, 2008). The method is often used in a standard knowledge testing in many domains such as The Test of English for International Communication (TOEIC), The Japanese Language Proficiency Test (JLPT), The Information Technology Professional Examination (ITPE), The Fundamentals of Engineering (FE) examination, etc. The generated test-forms from a test assembly can help preventing in cheating and answer copying since the forms can be provided differently for each examiner. However, concerns on a test assembly are that difficulty and properties of a testlet such as a test length and a variation of test type should be standardly consistent.

There are two applicable approaches in a test assembly, i.e. a manual approach and an automatic approach. At first, the manual approach (Theunissen, 1985) was proposed and used thoroughly in the beginning of a test assembly era. However, heavy responsible is on a hand of test administrators for managing and creating fair test-forms to be equivalently difficult and to meet required standard and objectives of testing. With burdens on human, the task is time-consuming and requires much manpower to complete a round of assembly. Unfortunately, a question pool that should be frequently updated leads to the often changes in redoing a test assembly. Thus, there were several researchers proposing an automatic approach in a test assembly. They applied statistical models (van der Linden et al, 2006; Veldkamp, 1999) to automatically assemble test-forms from a pool. Their major challenge is to generate nearly equivalent test-forms in terms of difficulty and length to remain fairness in testing. At last, recent works on a test assembly can yield a satisfaction result by exploiting famous optimization method such as Branch-and-Bound (Nemhauser and Wolsey, 1988), Bee algorithm (Songmuang, 2011) and Genetic algorithm (Verschoor, 2004).

Despite the successful test assembly in terms of consistent difficulty and length, the generated test-forms remain an issue as redundancy of content, especially in domain-specific knowledge tests such as Information Technology Professional Examination (ITPE). The test-form tentatively contains semantically similar or same questions which can cause unfairness in testing. Examiners may gain an advantage of a test if there are two or more similar questions that they can do, or disadvantage for vice versa. This issue happens because the existing methods do not consider underlying meaning and topic of questions in an automatic assembling process. From observation, several types of semantic redundancy are found, but the major types are those with similar texts of the same topic questions and the same topic questions with different text, respectively. The similar questions are undoubtedly not preferable since the test-form containing them is not equivalent in difficulty to those without them and may lead to unfair and unqualified testing; Unqualified testing is a serious issue that leads to voiding of test results and wasting of testers' opportunity and resources. The issue is scalable regarding a number of questions in a question pool; the more questions in the pool, the more possibility for a test assembly to create test-forms with similar questions. Ordinarily, a question pool should be large for question variety; hence, a possibility of having similar questions is apparently high. In this work, we aim to solve the semantic redundancy problem using semantic-based similarity to minimize putting together similar questions in the generated test-forms from an automate test assembly task.

Background and Literature Review

Automated Test Assembly

An automated test assembly (ATA) is a common method to create test forms for a proficiency examination (Van der Linden, 2005). Several test forms are automatically assembled from a question pool in a concern of equivalent difficulty and length distribution. Generally, a task of developing ATA has problems including computational complexity and constraint satisfaction (Ariel, Veldkamp, and Breithaupt, 2006; Armstrong, Belov, and Weissman, 2005; De Jong, Steenkamp, and Veldkamp, 2009). The former problem is a problem from combinatorial optimization in selecting many items from a whole set, namely too many questions in a question pool for exponentially combining into many test-forms in a test assembly. The latter problem is how to select questions to meet required constraints which are mostly consistent difficulty and test length.

In automation, many approaches have been used to solve the problems. The approaches are the use of combinatorial optimization technique, machine learning technique and information-provided method. The first problem is solvable with the existing combinatorial optimization techniques such as Branch-and-Bound (B&B; Garey and Johnson, 1979) and Monte Carlo (Belov and Armstrong, 2008) method. The methods provide an optimization in computation by effectively reducing or focused search space or guessing good paths to decreases complexity in computation. The second problem however requires more information of each question to be assembled according to the set constraints. The methods to solve the second problem therefore apply the information in an assembling process such as Maximum Priority Index (MPI) and genetic algorithm (Verschoor, 2004).

Kuhn and Keifer (2013) proposed the Optimal Test Assembly in practice for the Austrian Educational Standards Assessment in Mathematics. In their work, the test-forms were generated using two consecutive procedures, i.e. starting at the item block level and continuing at the item level. In the first step, a partially balanced incomplete item block design was generated using simulated annealing. The incomplete item block then were filled with items (questions) using mixed-integer linear optimization. The method was reported to be used in practice and resulted in generating a test for Austrian Educational Standards Assessment in Mathematics.

Wang et al. (2017) proposed using Maximum Priority Index (MPI) suggested by Cheng and Chang (2009) to implement an automated test assembly for a large-scale Chinese proficiency test. The ATA consists of three main components which are 1) item pool preparation, 2) parallel form generation, and 3) the quality evaluation and test forms output. To meet test requirements, the key process is the second component where difficulty of each question is calculated in maximization based on computed based on classical test theory called a reliability-index-distance defined by Armstrong et al (1992). The generated 50 test-forms are then evaluated for quality and resulted to be acceptable regarding assigned constraints. Moreover, the time to generate the 50 test-forms is short in which indicates the solution to computational complexity problem.

From above-mentioned works, existing ATA systems finely perform their task in aiming to solve complexity from combinatorial optimization and fulfilling requirement constraints. Those works yet mentioned on similarity of questions in a generated test-form which can lead to unfairness in examination. A test with semantically similar or same questions is capable to be exploited by examiners to score higher than those with the test-form not containing similar questions. This issue thus should be solved to maintain fairness in examination. Recently, the similarity of questions in ATA was mentioned by Luantangsrisuk et al (2016). They proposed to apply unigram term frequency and inverse document frequency (TF-IDF) to detect similarity of terms appeared in a question for minimizing amount of similar questions in bee-algorithm based ATA. Their experimental results indicated that the method could significantly reduce a number of similar questions in generated test-forms. However, their work focuses on the term similarity, namely words with mostly same characters. In a natural language, completely different words can be synonymous or closely related. Hence, it would be better to additionally cover to those synonyms and terms with related meanings. In this work, the aim is to handle semantically similar terms in question to prevent semantic redundancy in ATA.

Materials and Methods

In this work, we aim to include semantic similarity measurement into Bee algorithm based automated test assembly to reduce a chance to group paraphrased questions into a test-form. The scope of this work is to invent a newly designed method applicable to the frequent used test assembly to enhance its ability to handle similar question assembling. Additionally, the invented method is designed to not interrupt the ability in reducing computational complexity of assembly. There are two main components in this system, namely Bee-algorithm part and semantic similarity calculation part. An overview of the proposed system is illustrated in Figure 1.

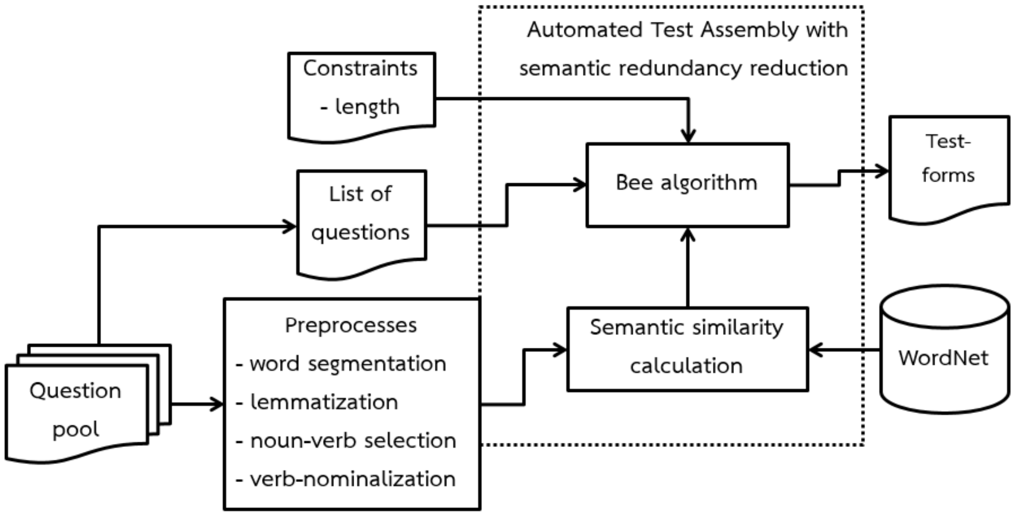

Figure 1. An overview of the automated test assembly with semantic redundancy reduction

1) Bee algorithm for Test assembly

Bee algorithm (Luantangsrisuk et al, 2016) is an optimization method that imitates foraging activity of honey bees in nature. The honey bees are split into two groups as scout bee and following bee in their honey-foraging activities. In Bee algorithm, the scout bees are responsible for randomly creating solution (global search) and evaluating a quality of the solution following an objective function (fitness). The following bees then follow the solution with the highest evaluation and perform random searches in nearby locations (local search). If the local search results in higher evaluation, the following bees continue local search until there is no higher evaluation. The process is continued until either an amount of limited cycle or assigned criterion is met.

The detail step of the Bee algorithm is on the following.

```
0. Parameter Initialization
    n = Number of employed bees
    m = Number of onlooker bees (m>n)
Iteration : Maximum iteration number
    $\alpha_j$: initial value of penalty parameter for j^th agent
EC-Length : Length of ejection chain neighbourhood
1. Initialize employed bees with GRAH algorithm
    $\sigma^i$ : i^th employed bee in the population
2. Evaluate employed bees
Fitness Function for minimization
    $\sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}\,x_{ij} + \alpha\sum_{j=1}^{m}\max\left\{0, \sum_{i=1}^{n} b_{ij}\,x_{ij} - \alpha_j\right\}$
bij = resource capacity of agent i is assigned to agent j ;
cij = cost of task i if  assigned  to agent j
xij = decision variable (xij=1, if task i is assigned to agent j; 0, otherwise)
3. Repeat
    Cycle = 1
    1. Number of Scout bees = 0,1*n
    2. For each Employed Bee
        a. Apply SHIFT Neighbourhood
        If fit(ShiftNeighbour)<fit(EmployedBee) then
            Employed Bee = Shift Neighbour
        b. Apply DOUBLESHIFT Neighbourhood
        If fit(DoubleShiftNeighbour)<fit(EmployedBee) then
            Employed Bee = DoubleShift Neighbour
        c. Determine probabilities by using fitness function
            $p_i = \frac{\Sigma\left(1/\mathrm{fit}_i\right)^i}{\mathrm{fit}_i}$
        d. Calculate the number of onlooker bees which will be sent to food sources
of employed bees, according to previously determined
    probabilities
        e. $N_i$= Number of onlooker bees sent to i^th sites = $p_i$*m
        $O_{ij}$: jth onlooker bee of ith solution (j=1,…,$N_i$)
        {$O_{i1}$, $O_{i2}$,…,$O_iN_i$}=EjectionChain($\sigma^i$)
      g. Calculate fitness values for each onlooker bee
        If the best fitness value of onlooker bees is better than the
        fitness value of employed bee, employed bee objective function
        is replaced with this onlooker objective function.
        If (min (fit(Oij))<fit($\sigma^i$) then $\sigma^i$= Oij
   3. Best Solution
      If fit(BestCycle-1)> Min(Fit($\sigma^i$))i=1..n then BestCycle= $\sigma^i$
      Else BestCycle= BestCycle-1
      Until (i=n)
   4. Scout bees
      a. Initialize scout bees with GRAH algorithm
      b. The worst employed bees as many as the number of scout bees in
      the population are respectively compared with the scout objective function.
      If the scout objective function is better than employed objective function,
      Employed objective function is replaced with scout objective function n. Else
      employed objective function n is transferred to the next cycle without any
      change.
   5. Cycle = Cycle+1
      Until (cycle =Iteration)
```

In this work, an objective function for Bee algorithm is a semantic similarity score of questions; thus, we can formulate the fitness using the following equation (1).

$$fitness = F(Q) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} semsim(Q_i, Q_j) \qquad (1)$$

where fitness is a result of an objective function. $F(Q)$ is an objective function, and Q is the set of question $\{Q_i, \ldots, Q_N\}$ in a test-form. N is a test length. $semsim(Q_i, Q_j)$ refers to a semantic similarity score between Q_i and Q_j.

In the initial step, scouting bees randomly find solution which is to choose a question matching a given constraint. All the randomly selected questions are then evaluated to find the cycle fitness. Afterwards, a calculation of probability to assign a following bee to some certain solutions is obtained using (2) as follows.

$$p_x = \frac{\frac{1}{f_x}}{\sum_{y=1}^{N}\frac{1}{f_y}}; x \in S \tag{2}$$

where *S* is a set of all solutions made by scouting bees, and *x* is a solution that is a subset of *S*. P_x is a probability of solution *x*. f_x refers to a fitness value of the solution *x*, and f_y is a fitness value of solution *y* while *N* is a number of solution.

For the last step, the solution made by a following bee is compared to the solution initially assigned by the scouting bee. If the evaluation made by the following bee is higher than the originated evaluation, the originated solution is replaced with the new highest one. The cycles are continued until the limit of cycles is reached or the specified criterion is met.

2) **Semantic Similarity Calculation**

To minimize semantically similar questions in a generated test-form, all content words in every question should be recognized. The content words in this work refer to only noun and verb as they are cores of meaning in a question while adjective and adverb are excluded since they play a role of a modifier to other words. The found nouns are to be lemmatized (removing a plural inflection or transforming a plural form to a dictionary form, such as “books $\rightarrow$ book” and “mice $\rightarrow$ mouse”) using NLTK tool (https://text-processing.com/demo/stem/).The found verbs however are normalized and transformed into a noun via nominalization to prevent mismatching.

In representing semantic relation, WordNet (Miller, 1995), which is a famous and frequent-used semantic resource, is exploited to find meaning relationship among terms. WordNet provides a set of synonymous terms (SynSet) and a hierarchical relation of the sets. The hierarchy identifies hypernym (more general meaning) and hyponym (more specific meaning) relation where the closer hierarchical level refers to closer in meaning and vice versa. The hierarchical structure of WordNet hence is treated as a light-weight ontology (Davies, 2010) to define meaning relation. An example of WordNet is given in Figure 2.

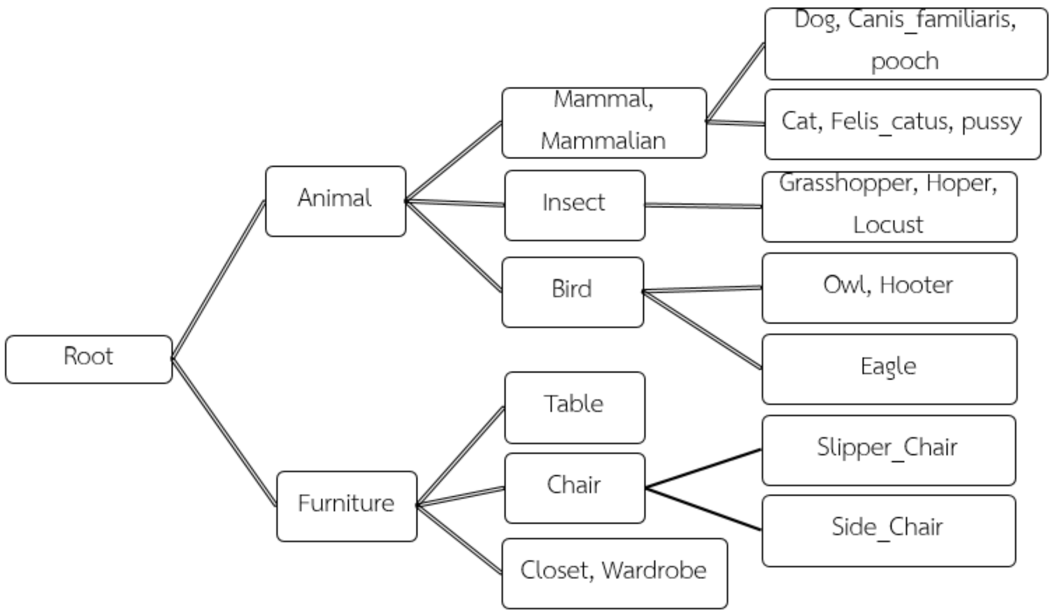

Figure 2. Hierarchical structure of terms used for calculating semantic similarity

From Figure 2, the concepts are in a hierarchical relation (is-a relation). The concepts in a block are synonym to each other as they have the very same meaning. The concepts in the same branch are closer in terms of related meaning while the concepts of the different branches are most likely semantically unrelated. For example, the term "Table" is much more semantically similar to "Chair" than to "Owl". The closer the concepts are distant, the more similar in meaning they are.

To calculate $semsim(Q_i, Q_j)$, all remaining words in questions are scored using (3) and (4) (Na Chai et al, 2016). Before further explanation, let's define notations as follows:

1) $len(c_1, c_2)$: the length of the shortest path from SynSet of c_1 to SynSet c_2 in WordNet. In the case of the concepts of the same SynSet, $len(c_1, c_2) = 0$.

2) $lso(c_1, c_2)$*:* the lowest common subsumer of c_1 and c_2

3) $depth(lso)$*:* the length of the path to SynSet from the global root concept while the depth at root is 1 and count onwards.

By using words in a question as a concept, the hierarchical structure of WordNet can be used to measure semantic similarity.

$$semsim(W_i, W_j) = \frac{2*depth(lso(c_1,c_2))}{len(c_1,c_2)+ 2*depth(lso(c_1,c_2))} \tag{3}$$

$$semsim(Q_i, Q_j) = \frac{\sum_{i=1}^{n} semsim(max(W_i,W_j))}{count(W_{total})} \tag{4}$$

where W is a word in a question, and n is a number of word while W_{total} is the total number of word in a question. In this work, the lowest similarity is a wanted result for fitness function in Bee algorithm since we aim to reduce the same semantic meaning within questions in the same test-form.

Experiments, Results and Discussion

Experiment Setting

A dataset used in this experiment was a set of English questions collected from Information Technology Professional Examination (ITPE) used during fiscal year of 2007 to 2010. There were 400 questions in total (100 questions for each year). By manually counting, there were 3755 pairs that were semantically redundant among all pairs of questions. For assembly setting, an amount of questions per a test-form was aligned to 50 questions and 4 test-forms. Same questions were allowed to be used in different test-forms, but not in the same test-form. Cycle iteration was set to 50 and 100 iterations, and 20 bees were set for bee population. For WordNet resource, a noun file of WordNet 2.0 was used for semantic similarity calculation. The measurement in this experiment was a counting of redundant question in the same generated test-forms. A comparison between using of semantic similarity score and without using was recorded to differentiate proficiency of applying semantic similarity for reducing redundant questions in an ATA task.

Results and Discussions

The counting amount of redundant questions in the same test-form is provided in comparison between using semantic similarity and without semantic similarity in Table.1.

Table.1 A comparison result in redundant question counting

	Iteration	1#test-form	2#test-form	3#test-form	4#test-form	Avg.
Without Semantic Similarity	50	11/50	12/50	14/50	13/50	12.5/50
	100	12/50	10/50	13/50	14/50	12.25/50
With Semantic Similarity	50	2/50	2/50	4/50	3/50	2.75/50
	100	2/50	0/50	2/50	2/50	1.5/50

The results indicated that the difference between using semantic similarity and without using it was clearly distinguished. The test-forms generated from using semantic similarity as objective function obtained much lower number of redundant questions for approximately 10 questions in average than without using it. The results proved that the proposed method could help in minimizing redundant questions in an ATA task. In other aspects, a test length of assembled test-formed was in acceptable range with around 15-16 pages in a printed version.

In observing the results in detail, the remaining redundant questions from ATA using semantic similarity were the questions with scarcely seen technical terms, which do not exist in WordNet. Since this work mainly applies semantic information from a structure of WordNet in semantic similarity calculation, the proposed method could not prevent this issue. In fact, this work conducted experiment on ITPE questions in which contain information technology and computer related terms that mostly appear in WordNet. Thus, the results were satisfied. In other hand, if the terms in questions are not available in WordNet such as terms in nano-technology or physics, the proficiency of the system may suffer greatly. Therefore, it can be claimed that the performance of the proposed system heavily relies on informative and coverage of the used semantic resource.

Conclusions

This paper presents a method to develop an automated test assembly which can minimize semantic redundant questions in the same test-form. The proposed system exploits Bee algorithm to assemble questions from a question pool and manage assigned constraints with its capable optimization. Furthermore, a semantic similarity calculation is used to recognize meaning of terms in a question to prevent assembling similar questions together. WordNet is applied as a semantic resource for examining synonym and hierarchical relation of terms.

The experiments were conducted using 400 questions from ITPE of 2007 to 2010, and a comparison between applying semantic similarity in an objective function in Bee algorithm and not applying was studied. From experiments, the results indicate that the proposed method helps to reduce an amount of similar questions in the same test-form for more than 90% in average. Moreover, other required performances such as a test length and computational time are in acceptable range. The generated 50 questioned test-forms of a setting of 100 iterations and 20-bee population were the best for lowest number of semantically redundant questions.

References

Ariel, A., Veldkamp, B. P., & Breithaupt, K. (2006). Optimal testlet pool assembly for multi-stage testing designs. Applied Psychological Measurement, 30, 204–215.

Armstrong, R. D., Jones, D. H., & Wu, I. L. (1992). An automated test development of parallel tests from a seed test. Psychometrika, 57 (2), 271–288.

Armstrong, R. D., Belov, D. I., & Weissman, A. (2005). Developing and assembling the Law School Admission Test. Interfaces, 35, 140–151.ข

Belov, D. I. (2008). Uniform test assembly. Psychometrika, 73, 21–38.

Belov, D. I., Armstrong, R. D., & Weissman, A. (2008). A Monte Carlo approach for adaptive testing with content constraints. Applied Psychological Measurement, 32, 431–446.

Cheng, Y., Chang, H. (2009). The maximum priority index method for severely constrained item selection in computerized adaptive testing. British Journal of Mathematical and Statistical Psychology, 62, 369-383.

Davies, J. (2010). Lightweight Ontologies. In: Theory and Applications of Ontology: Computer Applications, 2010, 197–229.

De Jong, M. G., Steenkamp, J. B. E. M., & Veldkamp, B. P. (2009). A model for the construction of country-specific, yet internationally comparable short-form marketing scales. Marketing Science, 28, 674–689.

Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: W. H. Freeman and Company.

Luantangsrisuk V., Songmuang P., Kongkachandra R. (2016). Automated Test Assembly with Minimum Redundant Questions Based on Bee Algorithm. InSignal-Image Technology & Internet-Based Systems (SITIS), 2016, 652-656.

Kuhn, J., Keifer, T. (2013). Optimal Test Assembly in Practice: The Design of the Austrian Educational Standards Assessment in Mathematics. Zeitschrift für Psychologie, 201, 190-200.

Miller, G. A. (1995) WordNet: a lexical database for English. Commun ACM. ACM, 38(11), 39–41.

Na Chai, W., Ruangrajitpakorn, T., Buranarach, M., Supnithi, T. (2016) A Framework to Generate Carrier Path Using Semantic Similarity of Competencies in Job Position. Trends in Artificial Intelligence: PRICAI 2016 Workshops: PeHealth 2016. 89-100.

Nemhauser, G., Wolsey, L. (1988). Integer and combinatorial optimization. New York: John Wiley & Sons, Inc.

Songmuang, P. Ueno, M. (2011) Bees algorithm for construction of multiple test forms in e-testing. IEEE Trans Learn Technol. 4(3), 209–21.

Theunissen, T. (1985). Binary programming and test design. Psychometrika, 50 (4), 411–420. doi: 10.1007/bf02296260

Wang, T., Zheng, Y. Zheng, C. Su, Y. (2016). An Automated Test Assembly Design for a Large-Scale Chinese Proficiency Test. Applied Psychological Measurement, 40(3), 233-237.

van der Linden, W. J. (2005). Linear models for optimal test design. New York: Springer-Verlag.

van der Linden, W. J., Ariel, A., & Veldkamp, B. P. (2006). Assembling a CAT item pool as a set of linear tests. Journal of Educational and Behavioral Statistics, 31, 81–99.

Veldkamp, B. P. (1999). Multiple objective test assembly problems. Journal of Educational Measurement, 36, 253–266.

Veldkamp, B. P. (2002). Multidimensional constrained test assembly. Applied Psychological Measurement, 26(2), 133–146.

Verschoor, A. (2004). IRT test assembly using genetic algorithms. Arnhem: Cito.