Second Semester Mini-project
Optimal Winner for Combinatorial Auctions

Blazej Franciszek Jozefiak
email: msc48bfj@bham.ac.uk, student number: 0828340

University supervisory by Dr Xudong Luo
Industry supervisory by Dr Abhijit Sengupta and Dr Danica Greetham

School of Computer Science, University of Birmingham,
Birmingham, B15 2TT, United Kingdom

April 30, 2007

April 30, 2007

Abstract.

Internet offers a platform where various models of auctions have been employed to trade goods and services. This paper is a response to trade market that expects new e-commerce solutions for clearing auctions. The market place with a single buyer and multiple sellers offering the multiple items in multiple quantities on buyer demand was created. This problem was defined with Multidimensional Knapsack Problem (MKP) where a nature of the problem was transformed from the objective maximization to objective minimization in order to minimize the price payed for maximum quantities of goods. The auction winner was determined by genetic algorithm (GA). Evolutionary search strategy was supported with heuristic operators for repairing, mutating and crossing the individuals. The GA performance in market clearability was analyzed on large data sets.

Keywords: reverse combinatorial auction, multidimensional knapsack problem, genetic algorithms

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Agent-mediated e-commerce

Agent-mediated electronic commerce involves software agents acting on behalf of some or all the parties in e-commerce trading. Here a software agent is viewed as an encapsulated computer system, situated in some environment, that is capable of flexible autonomous action in that environment in order to meet its design objectives (Jennings R, 2003). The main reason for introducing the agents in e-commerce is an opportunity to trade faster, cheaper and more convenient for both parties. Additionally, replacement of human factor by computations leads to handling combinatorial negotiations rapidly with maximum return for parties.

1.2. Auctions in e-commerce

Auction theory studied and well presented by (Klemperer, 1999) is nowadays applied to e-commerce and e-business solutions that operate within the Internet by trading single goods as well as sophisticated combinations of flight tickets, radio frequencies, market shares and etc. The list of already existing online auctions is long and includes up to 2,600 auctions[1]. Auctions popularity can be found in effective methods that they offer for good and services allocation. Moreover, online auctions make the physical limitations of traditional auctions disappear (e.g. time, space and presence), and they provide millions of globally dispersed customers with more varieties of goods that can be selected within a flexible pricing mechanism (Jennings R, 2003).

There are four standard auction types that are widely presented in (Klemperer, 1999). These are the ascending-bid auction,the descending-bid auction, the first sealed-bid auction and the second sealed-bid auction and are well presented by (Ghebreamlak, 2005) and (Klemperer, 1999) which are an excellent introductory papers to auctions theory. The ascending-bid auction is also well known as the English auction where the price is rising until all other bidders have dropped off and the only one bidder remains. Either bidders can call a new price or the auctioneer what must be set before the beginning of the auction and cannot be changed. This is a popular type of antiques, artwork or properties auctions. The starting price in the descending-bid auction is very high and gradually lowered until the first bidder will accept the current price. This type of auction is called the Dutch auction and is applied to flower auctions in the Netherlands, to fish

[1] http://www.internetauctionlist.com

market in Israel or to tobacco market in Canada. In the first sealed-bid all bidders submit their bids independently without a knowledge of other bids values. The bidder who offers the highest value wins and pays his bid. It is applied to mineral rights in government-owned land auctions as well as in the procurement between competing contractors. In the sealed second-bid the bidder who offers the highest value wins but in contrary to previous auction type it pays the second highest bid. Bids are made independently without a knowledge of other bids values.

Auctions can be performed in sequential and parallel order for submitting bids (Sandholm, 2002c). In this paper a background knowledge is given for understanding its further implementations to real-life problems described in details by (Sven de Vries, 2001). The later paper gives a good overview of already existing applications of combinatorial auctions (CA) such as FCC radio frequency spectrum auction ran in 2001, online logistics consulting ran in 2000 by SAITECH-IN and stock market trading system offered in 1998 by OptiMark Tech. The paper of (Joni L. Jones, 2003) provides a realistic scenario for combinatorial auction where prime time television commercial advertisement time slots are the subject of auction sale.

For the purpose of developing a means of comparing and evaluating different types of agents representing different types of strategies, there were established several international competitions. These are RoboCup [2], RoboCupRescue [3] and TAC (Trading Agent Competition) [4].

Trading Agents Competition (TAC) was designed and lunched for the first time in the 2000. In order to compare different agents designs the competition is a set of 28 parallel auctions that are too complex for analytical derivation of optimal bids collections. For that reason TAC became a model description for market game using trading agents. Competing in TAC agents acquire for their clients travel goods in three different markets buying flight tickets, booking hotel rooms and entertainment tickets. The competition structure,technology used and description of rules are well presented by (Wellman M, 2001) and (Jennings R, 2003). (Ioannis A. Vetsikas, 2003a) and (Ioannis A. Vetsikas, 2003b) study the design of autonomous trading agents on example of *White Bear*, the TAC 2002 winner while (Jennings R, 2003) presents the *SouthamptonTAC* the one of the most successful participant of TAC

[2] An international project that uses football as a central topic, see `http://www.robocup.org` for more details (Jennings R, 2003)

[3] A new research domain which targets search and rescue in large scale disasters (such as earthquakes), see `http://www.r.cs.kobe-u.ac.jp/roborescue/formoredetails` (Jennings R, 2003)

[4] For more details see `http://tac.eecs.umich.edu/`

in 2001 and 2002. Additionally, these papers discuss a methodology for deciding the bidding strategy for agents participating in simultaneous auctions.

1.3. Combinatorial auctions

Combinatorial auctions (CAs) are those in which bidders can place bids on combinations of items, called "packages", rather just individual items (Cramton P, 2005). The advantage of CAs over previously mentioned auction models is ability given to bidder to express the preference that is especially important when item are complements. When the set of items has greater utility than the sum of utilities for the individual items (e.g. pair of shoes).

This paper discuses trading goods on the market place that is modeled by the reverse combinatorial auction. In general, the bidder is a buyer and auctioneer is a seller who maximizes its profit in combinatorial auction (CA). In reverse combinatorial auction an auctioneer buys and bidders sell where auctioneer minimizes its cost. Further details of this particular auction are provided in sections of *Scenario* and *Problem Definition*.

CA definition (Joni L. Jones, 2003) and (Nisan, 1999) specify four distinct features that define the combinatorial auction. These are as follows 1) how bids will be expressed, 2) how the winners will be determined and the allocation of items to each, 3) how payment for the goods will be determined, and 4) bidding strategy. This paper focuses on the first two features and these are defined in the section of *Problem Definition*.

In first feature, each bidder must be able to express a bid in combination of items. Since there are an exponential number of combinations the bidding language determines how much auctioneer is willing to pay for each possible combination of items. This topic is modeled with mathematical approach in section of *Reverse combinatorial auction problem definition*. Next key feature in CA definition is collection of items. Demanded by auctioneer items must be collected from the bidders. Collection process attempts the optimization of target function that is auctioneer cost minimization of collected items. This topic is modeled in section of *Winner Determination Problem Definition*.

CA allows bidders to submit a single bid for collection of multiple items. In CA bids are combinations of items that cannot be auctioned partially. This allows bidders to express complementarities between items instead of having to speculate into an item's valuation of the impact of possibly getting other, complementary items (Sandholm, 2002d). The need for this type of auction is apparent in situation exhibiting complementaries where the valuation of a combination of goods differs from the sum of the individual items value (Joni L. Jones, 2003). CA does not require the lookahead techniques as it takes place in sequential and parallel auctions what enhances its computational performance. Instead of that the auctioneer needs to determine the winner or winners.

This project focuses on agent-based market model with n sellers and one buyer where no agent is both a buyer and seller. The marketplace is considered as a multi-unit combinatorial reverse auction where multiple units of multiple items are traded simultaneously. Combinations of bids are treated as bundles combinations. Recent paper works (Jennings, 2002) proves that atomic combinations of bids can limit the potential profit available to the auctioneer; auctioneer may find it beneficial to accept several bids partially.

CA procedures There are different mechanisms for running the CA. The most popular one is Vickery-Clark-Grove (VCG) procedure where bidders report their evaluations for all packages; items are allocated efficiently to maximized total value. The Iterative Combinatorial Auctions (ICA) is another CA model where bidders express their preferences by providing provisional pricing and allocation information what helps bidders to focus on options that are most relevant. There an alternative to VCG auction called Ascending Proxy Auction (APA) where valuation is submit to proxy agent that bids iteratively on the most profitable package. The Clock-Proxy auction allows to avoid computational complexity with useful private discovery and has already been applied to high-stake auctions in several countries (Cramton P, 2005).

Bidding languages There are defined eight bidding languages presented in literature. (Nisan, 1999) lists them in following order starting with Atomic bids, OR bids, XOR bids, OR-of-XOR bids, XOR-of-OR bids, OR/XOR-formula, OR bids with phantom items-called OR* bids and Applet bids. (Sandholm, 1999) and (Sandholm, 2002d) considers OR-bids, XOR-bids and OR-XOR-bids as the examples of protocols for CA. These protocols are the sets of rules over the bids that characterizes auctions as the rule-based. The support the winner decisions supplying with complementarity between items.

In OR Bidding Language, each bidder can bid on combinations of indivisible items that are joined with nonexclusive OR. The OR bidding language focuses on settings where combinatorial bids are introduced to capture synergies (complementarities) among items (Sandholm, 2002d). This means that any number of bids can be accepted and provides the constraint that every item is allocated at most one bid. The winner determination problem is concerned of selecting these bids that would minimize the bid prices that buyer must pay for required items.

XOR Bidding Language is the responses to real-world settings where local sub-additivness can occur. Each bidder can submit multiple bids (each bid including a set of items and a price) but only one bid from any given bidder can be accepted; one XOR disjunct can be submitted; the language is expressive enough to represent any mapping from combinations of items to process (Sandholm, 2002d). XOR bidding language adds another constraint that no two bids from the same bidder can be accepted what reduces the search space. In OR-of-XORs Bidding Language allows for submitting multiple number of XOR disjuncts which are combined with OR, this provides a full expressiveness and convenience of using CA.

Winner Determination Problem Auction can be described as the collection of bids where the auctioneer's role is to select the best one. Collections of bids determine a type of auction while rules under which an auctioneer makes decision are known as Winner Determination Problem (WDP). This paper focuses on the bundle of combinatorial bids that determines a reversed combinatorial auction problem (RCAP). The role of the seller is to sell the items and get the highest possible payments for them while auctioneer's role is to acquire the items at the lowest possible price. (Ioannis A. Vetsikas, 2003a) analyzes different strategies for achieving equilibrium in context of TAC. In that example WDP is characterized by chosen strategy that is context dependant and changes in case of flight tickets, hotel rooms and entertainment tickets bookings. Virtual trading markets are distributed markets and for that purpose (Sandholm, 2002c) replaces the centralized auctioneer with distributed one. This paper describes distributed search-based solutions as well as negotiation-based approach. It explores the CAP by analyzing different search algorithms that generate feasible allocations of bids in order to propose an optimal solution for the real-life CAP approach. An interesting WDP approach was proposed by (Joni L. Jones, 2003) where the constraints were applied to auctioneer that in consequence would allow to model real-life business factors such as delivery time, distance, quality considerations, substitutability of numerous products as well as other business needs. Rule-Based Combinatorial Auction proposed in this paper considers XOR and OR expressiveness languages for an auction.

Computational complexity This project studies clearing problem for reverse combinatorial auctions that is concerned on applying a market clearing algorithms that match single buyer with multiple sellers. Further works are aimed on determining the winner who clears multi-unit and multi-item combinatorial reverse auction. The clearing problem in a combinatorial market is NP-complete and inapproximable; markets where is only one unit of one item for sale are trivial to clear (Sandholm, 2002b). A computational complexity of the clearing problem depends on several factors that determine the market. These are as follows a number of items on the market (single/multiple), a number of units on the market (single/multiple), a market pricing scheme (non-discriminatory/discriminatory pricing) as well as the shape of supply/demand curves which is not applicable to this project. However, markets with price wise linear curves are clearable in polynomial time under non-discriminatory pricing, but NP-complete to clear under discriminatory pricing (Sandholm, 2002b). Computational complexity is related to sequence of bidding in the auction. In a sequential auction, items are auctioned one at a time. If a bidder has a preferences over bundles, that is a combination of items, then bidding in such auctions is difficult; to determine its valuation for an item, the bidder needs to guess what items it will receive in later auctions; this counters peculation introduces computational cost and other wasteful overhead (Sandholm, 2002d). In parallel auction the items are open for auction and bidders may place their bids during a certain time period.

2. Problem Definition

2.1. Scenario

The aim of this section is to present a scenario of reverse combinatorial auction that is a subject of this project. This include auction inputs, expected outcome and done assumptions what results as an introductory part to the problem definition.

Consider the artificial farm F that is a major milk producer in the county. It supplies the large retail stores with whole milk, skimmed milk and semi-skimmed milk in the total number of 10000 pints daily. The milk production is dependant on following inputs as petrol, water, plastic bottles of 1/4 pint, pint, 2 pints and sugar. All these inputs are supplied by large number of sellers while most of them are oriented on milk farms and to be more competitive each of suppliers offer a wide range of products starting from petrol and ending with plastic bottles.

The farm F organizes an auction quarterly to refill its resources. Bidders are familiarized with inputs and their quantities that are demanded by the farm. The farm runs an reverse combinatorial auction playing a role of a single auctioneer who calls for m number of inputs in multiple units that are supplied by n number of bidders. A bid is a boundless package of items at single price and its items cannot be accepted partially.

The objective is to choose either a single bid or number of bids that supply all called items with the quantity equal or greater that demanded one at the minimum cost.

The first assumption is that each bid is considered to come from a new bidder. The second assumption is that a number of overlapped units in accepted bids is not essential and does not result in additional costs.

2.2. Reverse Auction Clearing Problem

The objective of this report is to present an approach for finding an optimal winner of the reverse combinatorial auction. The following section formalizes the clearing problem in multi-item, multi-unit, reverse combinatorial auctions. The auction deals with m-number of input items : $1, 2, ..., m$, supplied by n-number of bidders $b_1, b_2, ..., b_n$ and one auctioneer who has a demand $(c_1, c_2, ..., c_m)$ where c_j is the quantity of the input item j that the auctioneer wants. The bidder b_i makes an offer of the w_ij quantities for the input item j and $w_ij = 0$ if a bidder b_i does not provide the input item at all. The offer made by bidder b_i is associated with the price p_i. The bidder b_i's offer is boundless and cannot be accepted separetelly.

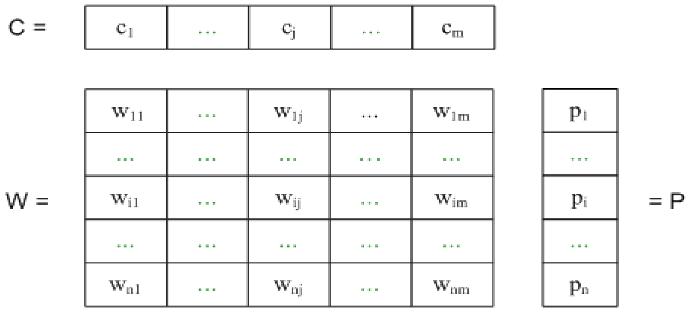

Figure 1. MKP problem definition

For example, lets imagine the auctioneer's demand for 3 input items in following quantities of $2, 3$ and 4. The bidders b_1 and b_2 offer a packages $b_1 = (2, 0, 1)$ with a price of $p_1 = 9$ and $b_2 = (7, 6, 0)$ with a price of $p_2 = 13$. The package b_1 is composed of demanded items 1 and 3 in quantities of $w_{11} = 2$ and $w_{13} = 1$ at a total price of 9 and the package b_2 is composed of demanded items 1 and 2 in quantities of $w_{21} = 7$ and $w_{22} = 6$ at a total price of 13. Bidder b_1 does not provide an item 2 $w_{12} = 0$ while the bidder b_2 does not provide an item 3 $w_{23} = 0$.

The reverse auction clearing problem is to find an allocation of bids that satisfies the constraints and maximizes the auctioneer's revenue. The auctioneer's revenue is defined by the fitness function that is the sum of prices to be paid for allocated in winning set of bids. The objective is to minimize the fitness function value as presented below. The minimization must be done with a subject to constraints.

$$min \sum_{i=1}^{N} p_i x_i, i = 1, ..., n, \tag{1}$$

subject to

$$\sum_{i=1}^{n} w_{ij} x_i \geq c_j, x_i \in \{0, 1\}, j = 1, ..., m \tag{2}$$

2.3. NP optimization problem

This section defines the NP optimization problem for the multi-item multi-unit reverse combinatorial auction with a free disposal. The section above has presented an example of auctioneer's demand and response of two bidders. To consider the reverse combinatorial auction as the NP optimization problem another bidder is invited. Third bidder b_3 offers a package of $b_3 = (1, 2, 3)$ with a price of $p_3 = 5$.

(Sandholm, 2002a) reduces the single-item multi-unit reverse combinatorial auction problem to *knapsack problem (KP)*. The standard KP is presented in eq.3 and 4. The first requirement for adapting the *knapsack problem* to the reverse auction is setting a *free disposal*, otherwise this approach would not be valid and another NP-optimization approaches would have to be adapted. For the purpose of KP adjustment to the multi-item multi-unit reverse combinatorial auction, there must be done two another major changes over the *knapsack problem*.

$$max \sum_{i=1}^{n} p_i x_i \tag{3}$$

subject to,

$$\sum_{i=1}^{n} w_{ij} x_i \geq c_j, x_i \in \{0,1\}, j = 1, ..., m \tag{4}$$

First of all, the multi-item multi-unit reverse combinatorial auction can be adapted to *Multi-dimensional Knapsack Problem* by specifying m-number of constraints as the auctioneer's demands c_j for each of j-th item, n-number of variables as the collection of bids b_is with specified coefficients w_{ij} as the quantities for each of j-th input item with associated to each i-th bid the p_i price. Let $I = \{1, ..., n\}$ and $J = \{1, ..., m\}$ with $c_j > 0$ for all $i \in I$ and $w_{ij} > 0$ for all $i \in I$ and $j \in J$. A well stated MKP assumes that $p_i > 0$ and $w_{ij} < c_j < \sum_{i=1}^{n} w_{ij}$ for all $i \in I$ and $j \in J$.

Each of m constraints is called a *knapsack* constraint and for that reason the reverse combinatorial auction is modeled with the*Multi-dimensional Knapsack Problem*. Other given names in literature are the *multi-constraint knapsack problem*, the *multi-knapsack problem* and the *multiple knapsack problem*. The special case corresponding to $m = 2$ is known as the *bidimensional knapsack problem* or the *bi-knapsack problem* (Beasley, 1998).

The MKP nature is oriented on the value-maximization problems while the reverse combinatorial auction is an example of the cost-minimization problem. For that purpose there is a second major change in the problem notation needed. The total number of offered units of input item j in all the bids is $T_j = \sum_{i=1}^{n} w_{ij}$. In this auction the auctioneer wishes to purchase at least $T_j - c_j$ units, at minimum possible price. Let S' be the set of bids that are winning bids in the reverse auction. Then, the remaining bids S that are not in S' form a solution to the knapsack problem. First of all, since the remaining bids have a total of at least $T_j - c_j$ units, the remaining units are at most c_j, and so the solution is knapsack feasible. Secondly, since the bids in S' provide $T_j - c_j$ at least possible price, the total price of remaining bids is largest possible subject to the knapsack constraints (Sandholm, 2002a). This is a problem reduction from NP-hard to NP-complete.

In other words, there are introduced the upper-bound constraints and lower-bound constraints. These are applied respectively for determining the subset of S and the subset of S'.

To summarize, there is a subset $S'$of winning bids that $min \sum_{i=1}^{n} p_i x_i$ and a subset S of loosing bids that $max \sum i = 1^n p_i x_i$ where $x_i \in \{0, 1\}$. The winner should be determined first by allocating the subset S subject to knapsack constraints and then the allocation of the remaining bids would form the winning set S with minimal price for supplying the input items.

According to (Pisinger, 1995), this is known as the *Multi - Constrained Knapsack Problem* which is a general Integer Programming Problem where all coefficients are non-negative integers.

2.4. Winner Determination Problem Definition

Winner Determination Problem (WDP) in reverse combinatorial auction can be formulated for a given set of bids by finding an allocation of items to auctioneer that maximizes its revenue. The bids are the expressions in a bidding language by which bidders report valuations for subsets of items (Lehmann and Sandholm, 2006a). The auctioneer's revenue is maximized by choosing an allocation that minimizes the cost of supplying all demanded items over all subsets of items that bidders offer.

Multi-Unit Multi-Item Combinatorial Reverse Auction Winner Determination Problem is to label the bid as winning or loosing so as to minimize the auctioneer's cost under the constraint that the auctioneer receives exact or higher number of the units of items that are required. This is a constraint relaxation by winner that determines a decision with a *free disposal* allowing the auctioneer to buy an extra units of item in order to minimize the cost.

A winning set S' of bids is the result of selecting the set S of loosing bids and excluding them from the set of all bids. This approach is known as *Multi -Constrained Knapsack Problem* and described in section before. The winner selects bids which items allocation maximizes the price paid for items in units specified by the constraint of knapsack capacity. The new WDP is defined as follows,

$$max \sum_{i=1}^{n} p_i x_i, x_i \in \{0, 1\} \tag{5}$$

s.t.

$$\sum_{i=1}^{n} w_{ij} \leq c_j \tag{6}$$

3. Algorithms for optimal winner determination in reverse combinatorial auction

This chapter focuses on search algorithms that would go through the search space and make a set of decisions. The tree search algorithms work by simulating all possible ways of making the decisions. The goal is to receive at the end of search the set of decisions that would prove the optimal solution.

(Lehmann and Sandholm, 2006b) discusses design and classification of search algorithms. This paper formulates the *Multivariate Branching* on question that include multiple variables. For the purpose of this work it is important that the algorithm can branch on the sum of values of a set of variables. The another important issue that determines the search algorithm design is the search strategy. There are several strategies widely described and applied to combinatorial auctions winner determination. (Lehmann and Sandholm, 2006b) presents DFS, DFS-BnB, A*, A* and best-bound, IDA*. [%for further description]

This report presents Genetic Algorithm for minimization the auctioneer's cost for supplying the quantities of demanded items. This algorithm solve the MKP under the constraint presented in section before.

3.1. Tabu Search for winner determination

The Tabu Search as an example of the local search algorithm was employed for determining the optimal winner of the reverse combinatorial auction at the initial stage of the project works. There were several reason for that such as the algorithm ability to start with a random solution that is evaluated and changed in order to improve it. The local search algorithms are a graph search space structure oriented where the elements are nodes and the edges between nodes creates a neighborhood. The local search algorithms iteratively move from one solution to another evaluating the fitness value until the stopping criteria are not satisfied.

The Tabu Search was applied for solving the problem while it tends to explore the search space regions more likely than other local search algorithms (e.g. Steepest Ascent). In comparison to Steepest Ascent, Tabu Search escapes from the basin of attention by removing the feasible bids from solution and allowing for decrease of fitness function value for a certain period of computation time.

Another important reason for applying the Tabu Search to the problem is short memory used by the algorithm. Tabu Search starts with an empty tabu list where adds iteratively points that are restricted for further solution. The points are released from tabu list after specified time that is called the tenure. Tenure time was modeled by the number of fitness function evaluations. In purpose of finding a set of bids the solution was modeled with tabu list that keeps set of feasible bids which improves the fitness value and cannot be selected more than once and are released after the tenure time. This allows the algorithm for exploring new regions of search space and is the main reason of applying the Tabu Search algorithm.

This is the pseudo code for the Tabu Search algorithm.
1. Add random bid to a current solution.
2. Remove the bid from solution if tenure was reached.
3.Check solution feasibility.
3a. If the current solution is feasible evaluate current fitness for current solution.
3b. Otherwise, void randomly added bid to current solution and go to 1.
4. If current solution improved the fitness keep it and go to 1.

3.2. Genetic algorithm for winner determination

A genetic algorithm (GA) can be understood as an *intelligent* probabilistic search algorithm which can be applied to a variety of combinatorial optimization problems (Beasley, 1998). It is hard to separate the filed of genetic algorithms from genetic programming and evolutionary strategies which seems to be integral parts and exist together. However, it is worth to mention the leading inventors of each sub-fields. Genetic algorithms were invented by John Holland in 1975 and used for combinatorial problems. Genetic programming was invented by John Koza and applied to synthesizing programs. Evolutionary strategies were invented by Schwefel and Rechenberg who used it for real-valued problems. Evolutionary programming was invented by Lary Fogel who applied it to finite state machines and by David Fogel in real-valued problems.

GAs are based on evolutionary process that takes place within biological organisms. During the course of evolution the organisms mutates and crosses resulting with offspring. In that way they evolve by adapting to environment conditions. The best adapted individuals survive whilst the others die what is called the natural selection. GAs follows this procedure by applying the operators of mutation, cross-over and selection to population of individuals on the course of finding the optimal one.

The crossover operation generates offspring from two chosen individuals in the population by exchanging some bits in the two individuals.

```
Generate an initial population;
 repeat
    Evaluate the fitness of parents;
    Apply genetic operators to produce children;
    Evaluate the fitness of children;
    Build a new population of parents and children.
 until a satisfactory solution has been found.
```

Figure 2. GA pseudo code

The offspring inherites some characteristics from each parent. The mutation randomly changes one or more bits in an individual. Mutation prevents local searches of the search space and increases the probability of finding the global optima. The selection operation chooses some offspring from survival according to predefined rules. This keeps population size within a fixed constant and puts good offspring into the next generation with a high probability (Tzung-Pei Hong, 2002).

(Michalewicz, 1992)mentions the following components that a genetic algorithm must include. They are,

- a genetic representation for potential solutions to the problem,
- a way to create an initial population of potential solutions,
- an evaluation function that plays the role of the environment, rating solutions in terms of their fitness,
- genetic operator that alter the composition of children during reproduction,
- values for various parameters that the genetic algorithm uses (population size, probabilities of applying genetic operators, etc.).

GAs presents an idea of gradual improvements to population fitness value. Members of population are individuals called chromosomes. Population size specifies number of individuals and their length. Each individual can be represented by integer or binary digits. Representation should increase individual compatibility for genetic operators.

4. Genetic Algorithm construction

4.1. Representation and fitness function

Number of individuals in population is various and described in details in section of experiments. Length of individual determines the solution length that is equal to number of bidders. Each individual in population represents a solution that is represented with binary bit where 1 is for included bids to and 0 is for excluded bids from solution. Index of binary bit referrers to bid number in a MKP problem matrix.

There are certain reasons for choosing the binary representation instead of the integer representation. First of all, the representation must be compatible with the genetic operators. The binary representation of solution was explained above. In the integer representation the solution would hold numbers of included bids. However, this would restrict the opportunity of including new bids to solution while mutation and crossover operators would not add to solution any new bids but only mix currently included ones. Binary representation allows the genetic operators for flipping $0s$ and $1s$ within the range of all bidders listed in the MKP problem matrix. In context of finding the winning set of bids the binary representation is more efficient that integer representation. First of all, the number of 1s and 0s included in individual affects the value of fitness function for the solution what can be manipulated by the genetic operators. Secondly, the binary representation allows the algorithm to include or exclude bids to and from the individual towards the global optimum. In context of this problem where the sequence of included bids to the solution does not affect the fitness function value the integer representation may affect the algorithm to get stuck in local optimum.

Fitness function F is a sum of prices to be paid for bids included in solution,

$$F = \sum_{i=1}^{N} p_i \qquad (7)$$

where N is total number of bidders and p_i is price to be paid for i-th bid.

4.2. Parent and offspring selection

GA starts with initial population of individuals. GA introduces population of parents and offspring. Each generation results with offspring that replaces parents according to selection operator. Parents and offspring are evaluated with fitness function and μ number of the best fitted parents and μ number of the best fitted offspring build new population.

4.3. Swap mutation

This is probabilistically-based mutation which swaps bits within the solution by including and excluding bids in solution. This is a one point swap mutation that is oriented on maximization of the price paid for the set of bids included in solution. Mutation swaps the bid that has associated the lowest price among all included in solution bids with the bid that has associated the highest prices among all excluded out of solution bids. In other words, the one point swap mutation replaces the 1 with 0 and vice versa in order to remove the cheapest bid in solution and to add the most expensive bid to solution.

The idea of one-point-swap mutation comes from the three-point-swap mutation presented by (Yao, 1997) for solving the cutting stock problem with a support of GA.

The presented one point swap mutation makes a decision probabilistically. For each individual in population there is the probability of included bids and the probability of excluded bids calculated separately. The probabilities Pr are calculated with following equation.

$$Pr = \frac{\frac{1}{\sqrt{p_i}}}{\sum_{i=1}^{L} \frac{1}{\sqrt{p_i}}}, \vee p_j \neq 0 \tag{8}$$

where L is the length of an individual and p_i is an associated price with i-th bid of the individual,

Equation (8) is counter-intuitive as it allows to bias towards the higher or lower value of probability. According to above equation the low priced bids are associated with high probability and the high priced bids are associated with low probability. The mutation algorithm follows these calculations to swap bids in order to maximize the price paid for set of bids included in solution. In other words, the one-point-swap mutation replaces the lowest price bid with the highest price one that was excluded from the individual.

Presented three point swap mutation is possible only with binary representation and proves its advantage against the integer representation of solution.

4.4. Uniform crossover

Uniform crossover was implemented for the offspring generation as the default crossover operator.

In the uniform crossover two parents have a single child. Each bit in the child solution is created by copying the corresponding bit from one or the other parent, chosen according to a binary random number generator [0,1]. If the random number is a 0, the bit is copied from the first parent, if it is a 1, the bit is copied from the second parent (Beasley, 1998).

The algorithm evaluates individuals of population with fitness function and chooses two parents - the best one and the worst one.

4.5. Constraints handling and repair operator

This application of GA to reverse combinatorial auction handles the constraint. The improvement of individuals fitness value must be followed with individuals feasibility to constraints. Constraints were specified in section of problem definition while the aim of this section is to describe handling and individuals repairing operators.

For each individual in population the feasibility function checks if the cumulated by the individual quantities of each input handle the constraints. Feasibility function returns a binary string of length equal to the number of inputs where 0s are for infeasible quantities of item and 1s otherwise.

There were different approaches applied to individuals repairing and were exercised for handling the constraints. They are described more in detail in section of experiments. Here is given the final repair function that is implemented to the GA for different number of runs. The repair function takes as an input the binary string returned by the feasibility function.

The repair function uses the look-up table that holds the weight-price ratios of inputs offered by each bidders. In other words, the weight-price ratio table is in the size of the input matrix, its every cell holds the result of the weight w_{ij} division by the price p_i according to the following formula.

$$WPR_{ij} = \frac{w_{ij}}{p_i} \tag{9}$$

Weight-price ratio rises with increase of weight followed by decrease of offered price and vice-versa.

For each infeasible input the repair function checks all bids excluded from the individual that are marked with 0s and finds the one with the lowest weight-price ratio. This strategy repairs the individual by adding 1 to the individual at the position of bid with the lowest weight-price ratio that is returned by the repair function for the infeasible individual. Presented strategy ensures that fitness value will be maximized.

```
1. Set the population size and selection rate.

2. Calculate WPR table.

3. Generate initial population.

4. Start algorithm until maximum number of iterations has not been
reached.

     4a. Evaluate the population fitness.
     4b. Uniform crossover operation.
     4c. One-point-swap mutation operation.
     4d. Constraints handling and repair function.
     4e. Selection operation.
```

Figure 3. GA pseudo code

4.6. Initial population

Initial population is generated randomly as the uniformly distributed random binary bits. The generated initial population is not feasible. However, there was a different approach applied for the initial population generation that is described more in detail in section of experiments.

4.7. Algorithmic outline

This section outlines the GA that has been developed to find the optimal winner for the reverse combinatorial auction. The pseudo-code for the algorithm is as follows.

5. Experiments

5.1. Problem generation

As far as I am aware at the moment of writing this report there are no publicly available data-sets for MKP that would fit to clearing problem of the reverse combinatorial auction. For the purpose of testing the effectiveness of the GA, there were data-sets generated independently. The idea for data-sets generation was adopted from (Beasley, 1998).

To generate data-set there must be specified the m number of inputs in auction and the n number of bidders making offers. These are the parameters determining the problem size. The MKP problem matrix consists of the weights w_{ij} as the quantities for inputs j offered by bidders i and of the associated to i-th bidder price p_i. The weights are the integer numbers drawn from the uniform generator $U(0, 1000)$. The values of price p_i are generated according to the following formula where α is a tightness ratio. The tightness ratio α correlates coefficients p_i and w_{ij} making the problem in general more difficult than uncorrelated problems.

$$p_i = \alpha * \sum_{j=1}^{m} w_{ij} \tag{10}$$

The MKP problem matrix is in a size of the w_{ij} weight matrix mxn plus the column of p_i price matrix.

The demanded quantities of inputs in this problem are generated according to the following formula.

$$c_j = \sum_{i=1}^{n} \frac{r_{ij}}{m} + 500q_i \tag{11}$$

where $j = 1, ..., m$ and q_i is a real number drawn from uniform generator $U(0, 1)$.

Problem data-sets determines the results that return the algorithm. There was noticed a close impact of the problem matrix to the feasibility of the finall individuals population. This is described in further section.

5.2. Initial population

In comparison to the previous section that presented the final approach to initial population generation this one discusses different approach that was tested within the process of the algorithm construction.

The main reason for trying different approaches was finding the final population of individual that were feasible. However, generation of population consisting of feasible individuals did not improve the algorithm final results. Resulted after 5000 generations by the GA individuals were infeasible in spite of applying the repair function.

For that reason the initial population consists of randomly generated individual that may be infeasible. And different techniques supported the feasibility improvements in this algorithm. There were found a close relation between results and problem data-set that influenced the algorithm and this is described in further section.

5.3. Mutation operator

The previous section of swap mutation presented the one-point-swap mutation that was implemented to the final algorithm. This section presents the different approach that was experimented within the process of GA construction but for reasons explained further was removed from the final version of the code.

Three-point-swap mutation was presented by the (Yao, 1997) for the purpose of GA construction that solved the cutting stock problem. In that paper an individual is decoded from the order list of all requested items and is represented as an integer string. Mutation of items in individual were affecting the fitness value of the individual. In context of finding the optimal winner for the reverse combinatorial auction the individual structure differs from the cutting stock problem one. Firs of all, the sequence of items that are bids included to the individual does not affect the fitness function value as it was in (Yao, 1997). An idea of probabilistically-based mutation showed at the three-point-swap was adopted to one-point-swap mutation in my algorithm where an individual is represented by the string of binary bits. Here, instead of bits sequence the number of 1s and 0s affects the fitness function value of an individual.

5.4. Constraints handling

There were different approaches to constraints handling exercised. Penalty functions with specified penalty factor value did not improve the fitness value of individuals in next generations. There were applied different penalty factor values within the range of $(0, 1)$ as the multiplicators of infeasible individuals fitness vales. For smaller penalty factors the products values were rapidly decreased for infeasible individuals. Static penalty did not promote the feasible individuals and as the value of penalty factor problem dependant the algorithm results did not benefited. Death penalty only advised when the feasible region is large. In context of the MKP problem matrix and randomly generated initial population of infeasible individuals the death penalty killed all population. Main reason was short number of feasible individuals and poor promotion of them by penalty approach.

Next step towards the feasibility and fitness improvement in population of individuals was implementation of repair function described in previous section. The another impact to the population quality had the MKP problem matrix described further.

Table I. GA parameters values

Parameter	Value	*Description*
n^{pop}	100	Population size
n^{max}	10,000	Max. number of individuals
^{max}	1,500	Max. number of individuals without improvement
μ	0.5	Mutation rate

6. Results

6.1. Results for large problems

Large data-sets were generated independently using the procedure presented above and adopted from (Beasley, 1998). The number of variables (bidders) n was set to 100, 250 and 300, and the number of constraints (input items) m was set to 5, 10 and 30 respectively. For each pair of $m - n$ combinations there were three different α tightness ratios applied, $\alpha = \{0.25, 0.5, 0.75\}$.

The algorithm terminates if termination criteria are reached. These are presented in table I which are as following the maximum number of individuals and the maximum number of individuals without improvement. Although, there are experiments over different number of individuals in population size table presents fixed values of population size and mutation rate which were found as the optimal ones.

The final fitness function value for feasible subset was measured over the different values of population size. Change of fitness function value over the population size is presented on the following figures 4 and 5. There were considered two extreme possibilities, the first one when number of bidders n equals to 100 and number of items m equals to 5 with a tightness ratio of α equals to 0.25 and the second one when number of bidders n equals to 500 and number of items m equals to 30 with a tightness ratio of α equals to 0.75. For both circumstances the mutation rate μ was set to value of 0.5 and the population size was varied between 10 and 300 individuals. The n^{pop} of 100 individuals was found as the reasonable size that would ensure the efficiency of computations and quality of results.

Finally, the fitness function value for the feasible and infeasible subsets was measured over the individuals' generation. Fitness function value was calculated for n^{pop} equals to 100 and μ equals to 0.5. According to *Genetic algorithm for winner determination*, on each of the graphs the continuous line presents the change of fitness function value

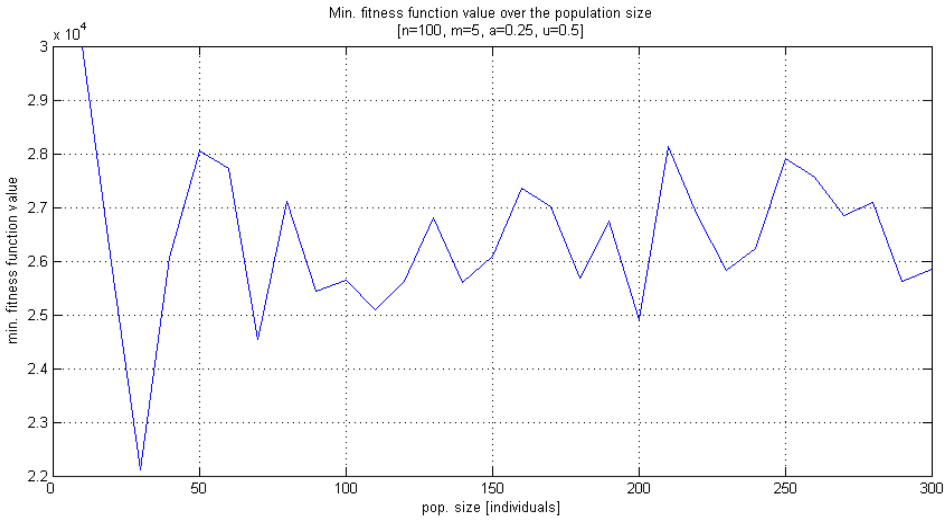

Figure 4. Fitness function value over population size for n =100, m =5, α =0.5, μ =0.5 and population size of 10 to 300 individuals.

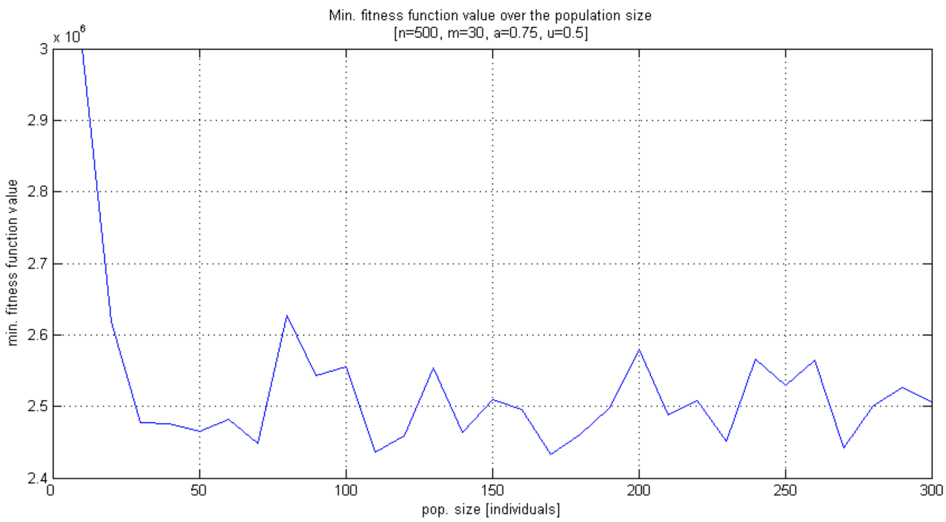

Figure 5. Fitness function value over population size for n =500, m =30, α =0.5, μ =0.5 and population size of 10 to 300 individuals.

for feasible subset of bids while the dashed line presents the fitness function value for infeasible subset of bids.

The problem has been solved using *LP relaxation* and GA. The results are shown in table.

Since the optimal solutions for the problem instances presented in the table are not known, the quality of results was measured accord-

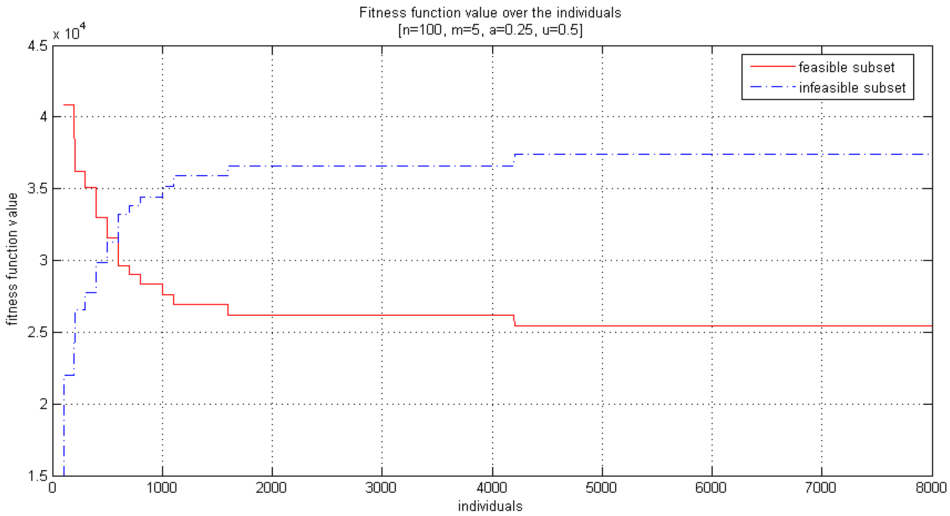

Figure 6. Fitness function value over generations for n =100, m =5 and α =0.25

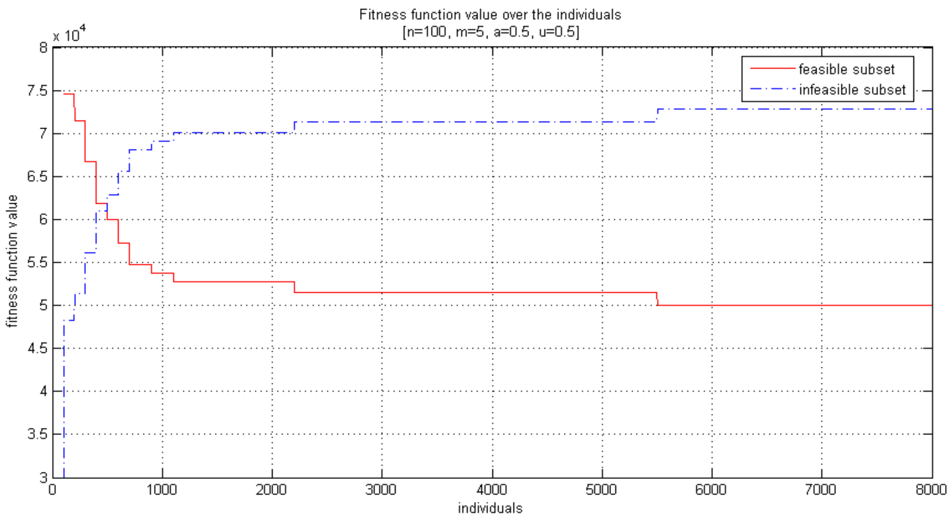

Figure 7. Fitness function value over generations for n =100, m =5 and α =0.5

ing to formula below with the percentage gap between the best result returned by GA and the best result returned by *LP relaxation*.

$$100 * \frac{optimalLPresult - optimalGAresult}{optimalLPresult} \tag{12}$$

The first three columns in the table indicate the problem size that is n number of variables, m number of constraints and the tightness ratio α where each problem structure contain 10 problem instances. The next

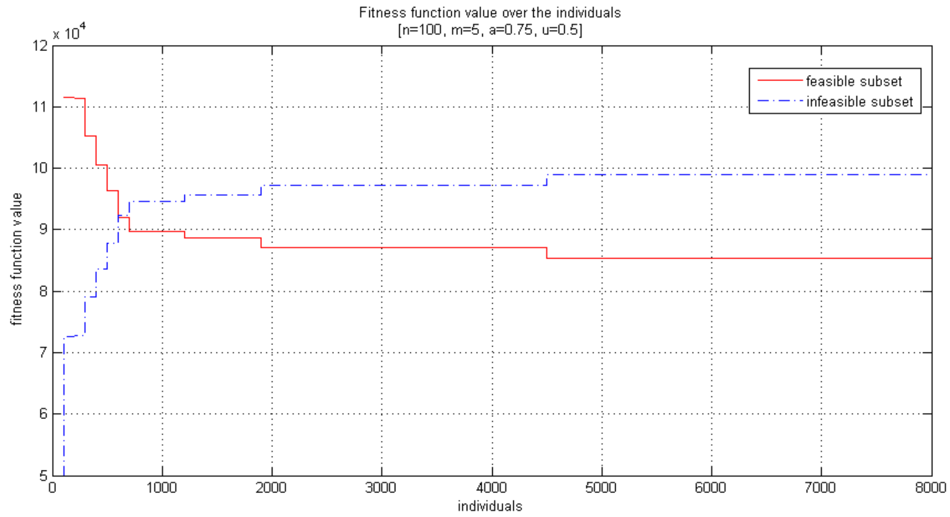

Figure 8. Fitness function value over generations for n =100, m =5 and α =0.75

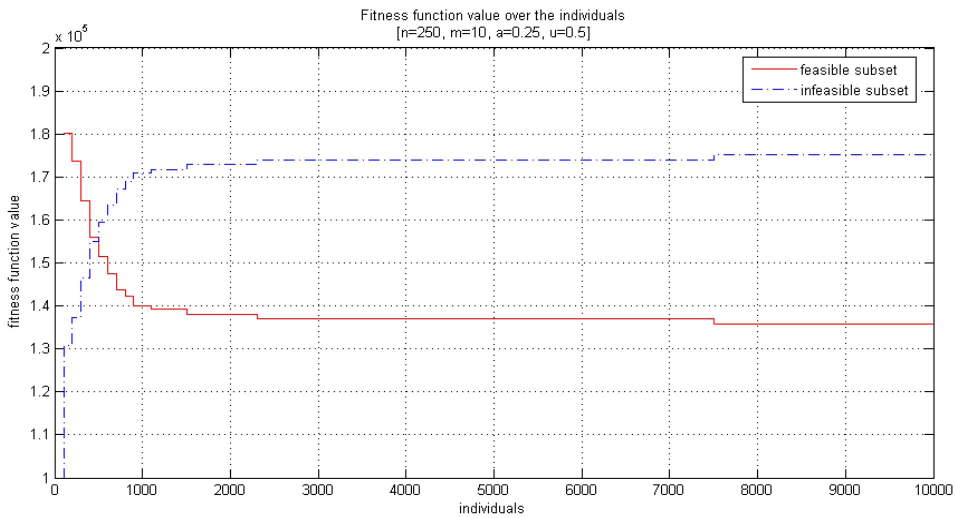

Figure 9. Fitness function value over generations for n =250, m =10 and α =0.25

column presents the average percentage gap over each 10 instances for each problem structure.

Table II presents the results of comparison GA with *LP relaxation* over the large data sets. The algorithms efficiency was expressed with %−gap where for all test instances the GA proves its efficiency. The larger a gap is, the result of GA are more efficient. In other words it means that the GA finds the cheaper allocation of bids that supplies the

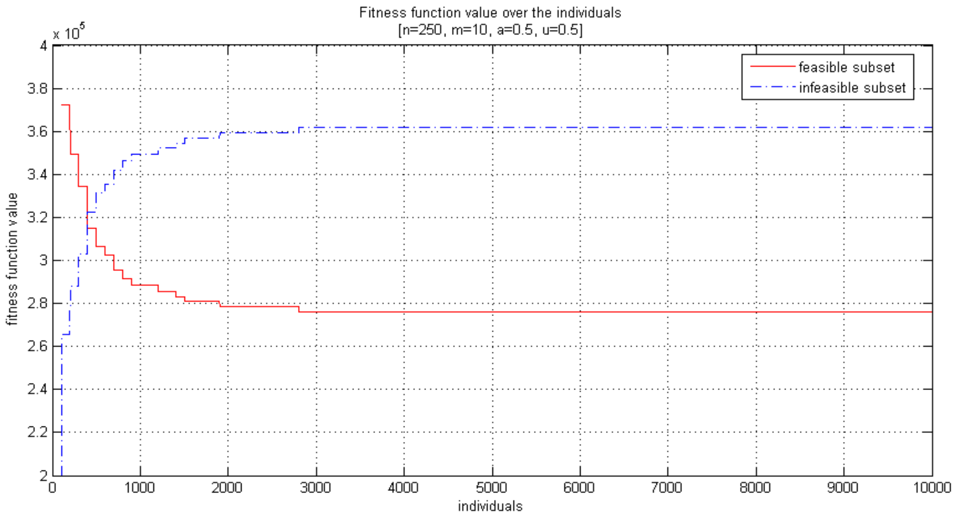

Figure 10. Fitness function value over generations for n =250, m =10 and α =0.5

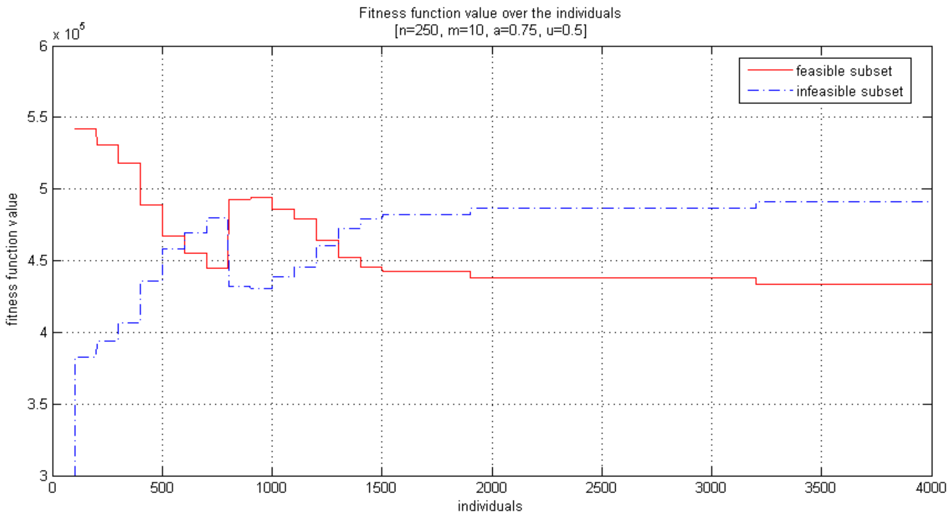

Figure 11. Fitness function value over generations for n =250, m =10 and α =0.75

input items requested by auctioneer. There can be noticed a difference in performance with a change of data sets size. The gap is getting smaller with increase of the size. The attempts of calculating the gap for $n = 30$ and $m = 500$ were unsuccessful while my implementation of LP relaxation built in algorithm in MATLAB environment was terminated when the maximum number of runs was reached.

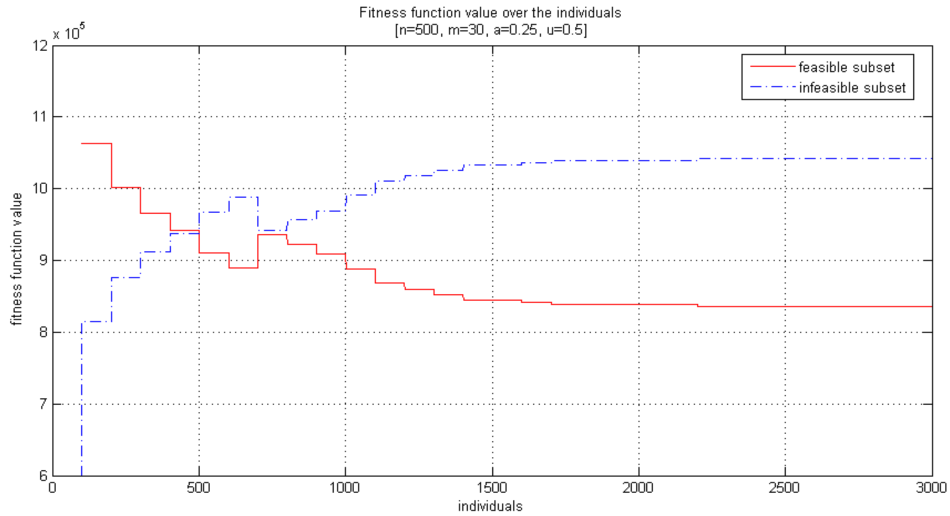

Figure 12. Fitness function value over generations for n =500, m =30 and α =0.25

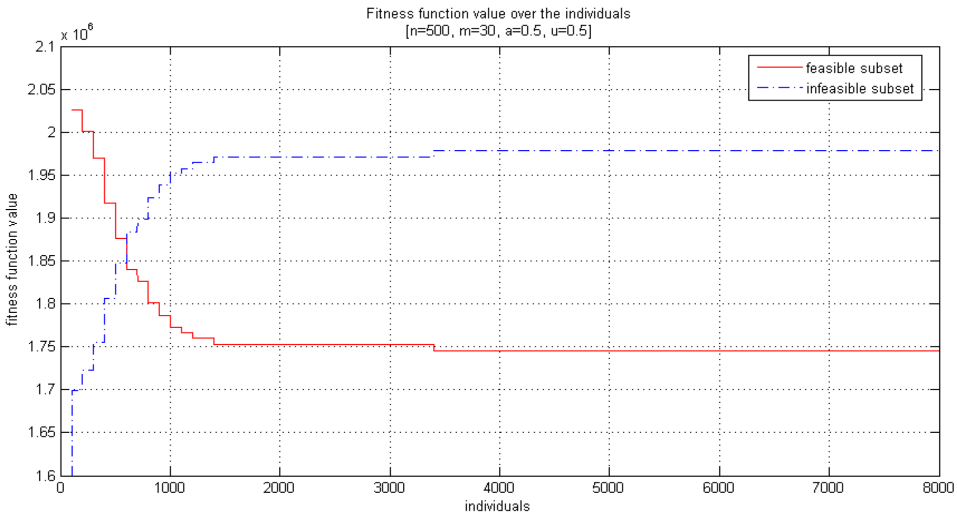

Figure 13. Fitness function value over generations for n =500, m =30 and α =0.5

7. Conclusions and Further work

7.1. GA biase towards revenue maximization

To conclude, this section describes an optimal winner determination by the genetic algorithm. There are four major steps that the algorithm follows in order to find the optimal allocation of bids that maximizes the auctioneer's revenue.

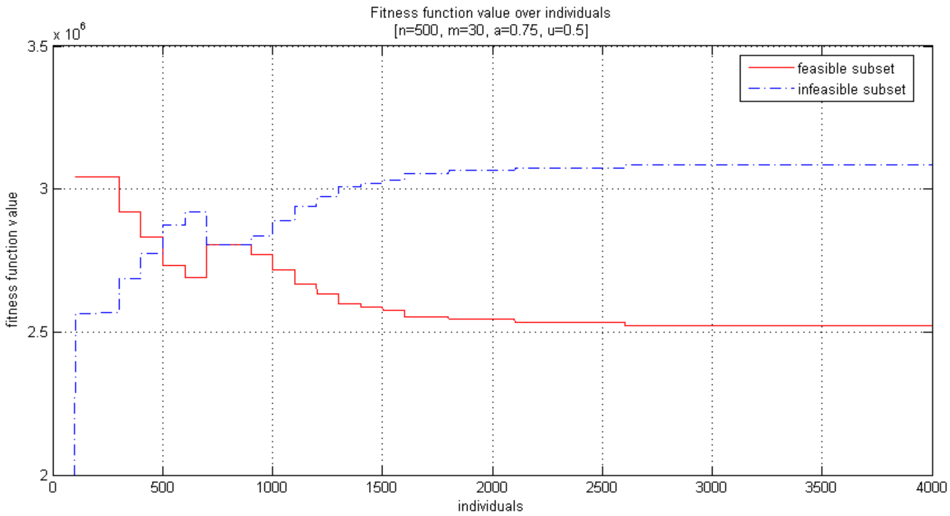

Figure 14. Fitness function value over generations for n =500, m =30 and α =0.75

Table II. Computational results for GA and LP relaxation

n	m	α	%-gap
100	5	0.25	59.90
100	5	0.5	54.15
100	5	0.75	52.39
250	10	0.25	9.63
250	10	0.5	29.91
250	10	0.75	22.39

According to the section of *NP Optimization Problem*, there is found an allocation that maximizes the price subject to upper-bounded knapsack constraints first and remaining bids form the allocation that minimizes the price subject to lower-bounded constraints.

Step One : One-point-swap mutation Probabilistically-based mutation operator swaps an existing in individual the lowest price bid with the new highest price bid.

Step Two : Feasibility function This operator ensures that the individual which does not meet a knapsack constraint would be repaired.

Step Three : Repair function This operator is based on the Weight-Price Ratio (WPR) look-up table. The repair operator removes from infeasible solution the highest WPR ratio bid in order to maximize the price paid for a subset of bids and to meet the knapsack constraint.

Step Four : Selection According to the value of μ selection rate, the selection operator replaces the worst ranked part of population with the best ranked part of offspring.

7.2. Further work

The construction of this GA algorithm for Winner Determination Problem was based on solving the MKP. Further works must be concerned on the algorithm application to retail and trade environments where more economical features should be considered during construction and testing phases. These are e.g. utilization of overlapping quantities.

This algorithm should be applied to negotiation-based environment where the final decisions about selecting winning allocations of bids would be made after running several stages of negotiations. This would involve building the procurement auction environment that handles processes of sending to and receiving by auctioneer the *RFQs* (request for quotation). For that reason the GA algorithm could be implemented to Vickrey-Clarke-Groves auction scheme as presented by (Debasis M, 2006).

There must be done improvements in the algorithm construction. First of all, it must handle the large-scale date where $n = 500$ and $m = 30$ with different α values of tightness ratio. Additionally, the genetic operators can be improved by applying LP or Lagrange relaxation techniques. For the purpose of improving the algorithm computational efficiency it can be combined with a local search techniques. Previous experience with Tabu Search implementation for reverse combinatorial auction may be valuable while combined with the existing GA would result with powerful memetic algorithm.

Finally, the algorithm code should be translated to object-oriented environment such as Java. This would improve the computational efficiency.

8. A statement of strategy information search

This section is to define types of literature used as the background for the report as well as for specifying the sources.

The collected papers can ba classified to following types:

- technical reports
- journal articles
- books
- PhD thesis

Geographical and language coverage Majority of papers are coming from Europe and United States of America. Language of papers is English.

Retrospective coverage It was efficient to search over 7 years back. However, some of the papers are older but still provide relevant knowledge.

Used tools and brief evaluation

For search purposes, the Science Citation Index (SCI) was applied. This allowed me to collect wide range of publications that are presented in the reference section.

References

Beasley, J.: 1998, 'A Genetic Algorithm for Multidimensonal Knapsack Problem'. Technical report, The Management School, Imperial College, England.

Cramton P, Shoham Y, S. R.: 2005, 'Combinatorial Auctions'. Technical report, MIT Press.

Debasis M, D. V.: 2006, 'Vickrey-Dutch procurement auction for multiple items'. In: *European Journal of Operational Research*, Vol. 2007. pp. 617–629.

Ghebreamlak, K. A.: 2005, 'Analaysis of Algorithms for Combinatorial Auctions and Related Problems'. Ph.D. thesis, Department of Mathematics, Uppsala University, Sweeden.

Ioannis A. Vetsikas, B. S.: 2003a, 'A Principled Study of the Design Tradeoffs for Autonomous Trading Agents'. Technical report, Computer Science Dept., Cornell University, Ithaca, NY 14853, USA.

Ioannis A. Vetsikas, B. S.: 2003b, 'WhiteBear: An Empirical Study of Design Tradeoffs for Autonomous Trading Agents'. Technical report, Computer Science Dept., Cornell University, Ithaca, NY 14853, USA.

Jennings, V. D. D. N. R.: 2002, 'Polynomial algorithms for clearning multi-unit single-item and multi-unit combinatorial reverse auctions'. Technical report, Intelligence Agents Multimedia Group, Department of Electronics and Computer Science, University of Southampton,UK.

Jennings R, H. M.: 2003, 'Designing a Successful Trading Agent : A Fuzzy Set Approach'. Technical report, Department of Electronics and Computer Science, University of Southampton, UK.

Joni L. Jones, G. J. K.: 2003, 'A Heuristic for Winner Determination in Rule-Based Combinatorial Auctions'. Technical report, University of Michigan Business School, 701 Tappan St D5210, Ann Arbor and Warrington College of Business, University of Florida, Gainesville, USA.

Klemperer, P.: 1999, 'A Survey of Auction Theory'. *Journal of Economic Survey* **13**(3), 227–286.

Lehmann, D Mueller, R. and T. Sandholm: 2006a, *Optimal Winner Determination Algorithms*, p. Chapter 12. MIT Press, 1st edition.

Lehmann, D Mueller, R. and T. Sandholm: 2006b, *Optimal Winner Determination Algorithms*, p. Chapter 14. MIT Press, 1st edition.

Michalewicz, Z.: 1992, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag.

Nisan, N.: 1999, 'Bidding and Alocation in Combinatorial Auctions'. Technical report, Institute of Computer Science, Hebrew University, Jerusalem.

Pisinger, D.: 1995, 'Algorithms for Knapsack Problem'. Ph.D. thesis, Department of Computer Science, University of Copenhagen, Denmark.

Sandholm, T; Subhashi, S.: 2002a, 'Market Clearability'. Technical report, Computer Science Department, Carnegie Mellon University, USA.

Sandholm, T; Subhashi, S.: 2002b, 'Optimal Clearing of Supply/Demand Curves'. Technical report, Computer Science Department, Carnegie Mellon University, USA.

Sandholm, T.: 1999, 'eMediator : A Next Generation Electronic Commerce Server'. Technical report, Department of Computer Science, Washington University, St. Louis, USA.

Sandholm, T.: 2002c, 'Algorithm for optimal winner determination in combinatorial auctions'. In: *Artificial Intelligence*. pp. 1–54.

Sandholm, T.: 2002d, 'eMediator : A Next Generation Electronic Commerce Server'. In: *Computational Intelligence*, Vol. 18. pp. 656–676.

Sven de Vries, R. V. V.: 2001, 'Combinatorial Auctions: A Survey'. Technical report, Department of Managerial Economics and Decision Sciences, Kellogg School of Management, Northwestern University, Evanston, USA.

Tzung-Pei Hong, H.-S. W.: 2002, 'Evolution of Appropriate Crossover and Mutation Operators in a Genetic Process'. Technical report, Department of Information Management, I-Shou University, Taiwan.

Wellman M, Wurman P OMalley K Bangera R Reeves D Walsh, W.: 2001, 'Designing the Market Game for a Trading Agent Competition'. *IEEE INTERNET COMPUTING* **3**, 43–51.

Yao, X.: 1997, 'A New Evolutionary Approach to Cutting Stock Problems With and Without Contiguity'. Technical report, School of Computer Science, University of Birmingham, Birmingham, UK.

Appendix

```
function [plotMin,plotMax,pFit,oFit,P] =
ga6(A,problemMatrix,pop_size)

%pop_size = 100;
u = 0.5; sol_size = size(problemMatrix,1);
%sol_size = 20;
max_range = size(problemMatrix,1);
bestPlot = 0; plotFit =  0;
plotMin = Inf;
plotMax = 0;
PrOf1 = []; PrOf0 = [];

for(i=1:size(problemMatrix,1))
    for(j=1:size(problemMatrix,2)-1)
        WPR(i,j) = problemMatrix(i,j)/
        problemMatrix(i,size(problemMatrix,2));
    end
end

P = randint(pop_size,sol_size,[1  0]);

cAi = 1; cBi = 1; while (cBi<1500 | cAi<10000)
    for i=1:size(P,1)
        pFit(i,1) = fitness2(A,problemMatrix,P(i,:));
    end

    for(i=1:size(P,1))
        if(pFit(i,1)>=plotMin(1,length(plotMin)))
            cBi = cBi + 1;
        else
            continue;
        end
    end


    %Plot
    [pv,pix] = min(pFit);
    plotMax(1,length(plotMax)+1) = pv;
    sol = P(pix,:);
    price = 0;
    for(i=1:length(sol))
        if(sol(i)==0)
            price = price + problemMatrix(i,
```

```
                 size(problemMatrix,2));
            else
                 continue;
            end
        end
        plotMin(1,length(plotMin)+1) = price;

        O = crossover(A,problemMatrix,P);

        O = mutate1PS2(problemMatrix,O);

        for(i=1:size(O,1))
             c=ones(1,size(problemMatrix,2)-1);
             c = feasibility2(A,problemMatrix,O(i,:));
             if(length(find(c==0))<=size(problemMatrix,2)-1)
                  sol2del = repair(WPR,c,O(i,:));
                  O(i,sol2del) = 0;
             else
                  continue;
             end
        end

        for i=1:size(O,1)
             oFit(i,1) = fitness2(A,problemMatrix,O(i,:));
        end

        [fP,inxP] = sort(pFit,'descend');
        [fO,inxO] = sort(oFit,'ascend');
        for i=1:(u*pop_size)
             tmpP(i,:) = P(inxP(i),:);
        end
        for i=1:(pop_size-u*pop_size)
             tmpO(i,:) = O(inxO(i),:);
        end

        P = [tmpP;tmpO];

        cAi = cAi + size(P,1);
end
```

```

function [O] = crossover(A,M,P)

for i=1:size(P,1)
     pFit(i,1) = fitness2(A,M,P(i,:));
end
```

```
[v, ix] = max(pFit);

[fD, ixD] = sort(pFit, 'descend');
[fA, ixA] = sort(pFit, 'ascend');

for(i=1:size(P,1))
    for(j=1:size(P,2))
        g = randint(1,1,[0 1]);
        if(g==1)
            O(i,j) = P(ixD(1),j);
        else
            O(i,j) = P(ixA(1),j);
        end
    end
end
```

```

function [c] = feasibility2(A,M,sol)

upper_bound = zeros(1,size(M,2)-1);
t = zeros(1,size(M,2)-1);

%sum of offered quantities
for(i=1:size(M,2)-1)
    t(1,i) = sum(M(:,i));
end

for(i=1:size(M,2)-1)
    upper_bound(1,i) = t(1,i);
end

%sum of quantities in solution
p = find(sol==1); u = zeros(1,size(M,2)-1);
for(j=1:length(p))
    for(i=1:size(M,2)-1)
        u(1,i) = u(1,i) + M(p(j),i);
    end
end

for(i=1:size(M,2)-1)
    if(u(1,i) < upper_bound(1,i))
        c(1,i) = 1; %valid bid
    else
        c(1,i) = 0; %invalid bid
    end
```

```
end
```

```
function [O] = mutate1PS2(M,P)

O = P;

PrOf1 = probabilityOf1(M,O); PrOf0 = probabilityOf0(M,O);

for (i=1:size(P,1))
    [v1,ix1] = max(PrOf1(i,:));
    [v0,ix0] = min(PrOf0(i,:));
    tmp = O(i,ix1(1));
    O(i,ix1(1)) = O(i,ix0(1));
    O(i,ix0(1)) = tmp;
end
```

```
function Pr = probabilityOf1(M,Pbin)

for(i=1:size(Pbin,1))
    tmp = 1e-06;
    for(j=1:size(Pbin,2))
        if(Pbin(i,j)==1)
            tmp = tmp + sqrt(M(j,size(M,2)));
        end
    end
    for(j=1:size(Pbin,2))
        if(Pbin(i,j)==1)
            Pr(i,j) = (1/sqrt(M(j,size(M,2))))/tmp;
        else
            Pr(i,j) = 0/tmp;
        end
    end
end
```

```
function Pr = probabilityOf0(M,Pbin)

for(i=1:size(Pbin,1))
    tmp = 1e-06;
    for(j=1:size(Pbin,2))
        if(Pbin(i,j)==0)
            tmp = tmp + sqrt(M(j,size(M,2)));
        end
    end
```

```
    for(j=1:size(Pbin,2))
        if(Pbin(i,j)==0)
            Pr(i,j) = (1/sqrt(M(j,size(M,2))))/tmp;
        else
            Pr(i,j) = 0/tmp;
        end
    end
end
```

```
function [sol2del] = repair(M,c,sol)

best = 0; sol2del = []; kp2rep = find(c==0);

for(kp=1:length(kp2rep))
    for(i=1:length(sol))
        if(sol(i)==0)
            ix = i;
            v = M(i,kp2rep(kp));
            if(v>best)
                best = v;
                sol2del = ix;
            else
                continue;
            end
        end
    end
end
```