A Novel Genetic Algorithm Selection Method and Implementation in IoT Domain

Kerem Aytaç
Marmara University,
Koç Digital R&D Center
Istanbul, Turkey
kerem.aytac@marun.edu.tr

Berkin Ağlarci
Yeditepe University
Istanbul, Turkey
berkinaglarci@gmail.com

Ömer Korçak
Marmara University
Istanbul, Turkey
omer.korcak@marmara.edu.tr

***Abstract*—In Internet of Things (IoT) domain, there are many NP-hard problems that are required to be solved efficiently by some meta-heuristic algorithms such as genetic algorithm. There are many genetic algorithm selection methods which aim to find the optimal chromosome by populating and varying them by some mutation and cross-over methodologies. Most of these algorithms process all of the genes of chromosomes into fitness function to decide whether to pass them to new generation or to apply some genetic operations or to abandon them. If chromosomes are ultimately long or fitness functions are difficult to compute, it will have a great overhead and will consume so much time. In this paper, a novel genetic algorithm selection method is proposed and promises some intelligence to keep these processes short and efficient by taking some remarkable risks. It is based on analogy of a football league with multiple groups and it uses the idea of partially processing the chromosome genes after adopting a decision mechanism based on Bayesian game theory. We implement the proposed selection algorithm to solve a complex problem in IoT domain and illustrate its performance compared to other generic selection methods.**

I. INTRODUCTION

IoT has many applications that solve troubles of daily life and industry. For example, Internet of connected devices is required if you want your coffee to be ready when you wake up in the morning, or if you want to be alerted when someone breaches into your home, or if you got a business and want to manage your customer queue efficiently, so on so forth. Typically, an IoT product generates very huge amount of raw data, all of which are valuable and should be processed effectively and in time-efficient manner.

To narrow down our domain, let us focus on some industrial areas. You have a quick service restaurant and you have many orders and some staffs to welcome them. Your staffs have some proficiencies on some product where some of them can be prepared quickly as they are experienced on it and some of them will be prepared slowly as they are inexperienced or non-talented on it. You have many criteria for orders such as their order type (Home delivery, In-Restaurant Delivery etc.). If it is home delivery, distance matters or priority customer orders should be prioritized (like loyalty program etc.), you got a shortage of a material which is required in order and this material should be retrieved by the assignee staff so that it will take too long to prepare it, so on so forth [1]… Any criteria here actually define your fitness function. Another industrial problem is Facility Location Problem [13]. It is also known as Location Analysis related with the optimal placement of facilities to minimize transportation costs while considering lots of factors and criteria like avoiding fuel waste, carrier capacity, customer demand, priorities like pre-paid orders, VIP customers, distances etc. Similar to previous problem, you may have lots of inputs which create your heavy fitness [3]. Such kind of problems are called multi-task generalized assignment problem which can be classified as NP-hard [2]. We have clearly seen that, in IoT area there are many NP hard problems that needs to be resolved in a computationally efficient manner. So, as a heuristic search algorithm, Genetic Algorithm is an appropriate approach to resolve these hard problems.

In this paper we extend a previous paper of ours named "IoT Based Smart Staff Allocator in Quick Service Restaurants" [1] which proposes a new genetic algorithm selection, namely "Bet Prediction Selection", which adopts a methodology not to iterate all genes in chromosome for a fitness function. It generates a population and creates a league. All chromosomes have matches like a football game between each other. In a match, two chromosomes start to get into the fitness function for their genes one by one. Every gene results a value after this function and is compared. If a gene is better than the other, then owner chromosome collects some point and a bet rate for likely being the winner in the end. If this rate reaches to a threshold or a decision point, match is over. Other genes will not get into fitness function. Because that selected chromosome has the higher rate and is winner-candidate. So like football bet games, we select that chromosome as the winner with a risk and error rate [1].

In a previous contribution of ours [1], bet rates were synthetic and the winner and loser were selected according to just a constant threshold value. Bet prediction selection was used as a population initialization method. Because, it just refined and filtered the population just before applying known and popular selection methods (like Elite selection etc.). Here in this paper, we improve this selection method and implement it as the main selection method where we abandon the popular ones, which is a promised future work noted in [1].

A game theoretic approach will be adopted to boost up the performance of the selection method. Game theory is an important tool that models strategic decision making of self-

interested agents [4]. It is a useful concept that can be adopted in many areas, including IoT domain. In [6], a game theoretic approach is used for an IoT-based automated employee performance evaluation. It proposes a model to industry in order to implement an award-penalty process for employees who behave good/bad. Behaviors and actions are collected with many IoT sensors, and all data create profiles and actions. Any bad or good activities can be detected with this way and classified as cooperate or not cooperate. Another decision branch is whether to award who cooperates or not and whether to fine who does not cooperate. Awarding is a payment, which is a negative activity on revenue, but awarding boosts employee to work well and gains more income. As a result, in a short time employees choose to cooperate so that industry gains more and more income. It actually turns into a win-win strategy [6].

Another work proposes a device-to-device cooperation framework for task allocation among any object in an IoT Solution. A node triggers the process by deciding to create a cluster of nodes and after that coordinating the allocation strategy. Here objects, which are capable of doing the same tasks, compete to get relevant remunerations. Therefore, they propose a game theoretical approach to maximize objects utility functions. Their game converges to a Nash Equilibrium Point using a bidding process [7].

In this paper we use game theory in radically different manner and we adopt an original approach where chromosomes are the players which decide whether to continue the match or withdraw during the selection process of genetic algorithm. This way we come up with reasonable and adaptive threshold values while predicting weak/strong chromosomes within a match, in contrast to constant threshold values adopted in the previous study [1].

This paper describes details of the proposed novel selection method for genetic algorithm, as well as its implementation to solve a generalized assignment problem in IoT domain. Next section describes a general problem definition and detailed explanation of selection method. Section III focuses on several tests and shows the efficiency of the proposed algorithm by comparing with existing methods. Section IV concludes the paper.

II. Background and Algorithm Proposal

In genetic algorithm, there are chromosomes which are built by genes. When the problem objects are extremely crowded, the chromosomes can be very long. Such as millions of customers demand some items from your side, and you have to welcome all demand and supply from your thousands of storage centers. Here you have millions of genes to fit in a fitness function, and in every iteration regardless of what selection method preferred, you have to compute the whole chromosome. Let's give an example for Roulette Selection. This selection method samples a population. It computes the fitness function for each individual and creates a pie such that the fittest chromosome covers the largest slice, on the contrary the worst chromosome covers the smallest slice. After that, a roulette spin occurs to randomly select a chromosome. The largest slice has the highest probability to be selected [8]. Another example is rank selection. Rank selection works similar to roulette selection. It computes all the fitness values and orders them in terms of values. According to order, they have the highest probability to be selected. So these entire computations cause an overhead [9]. On the other hand, Elite selection adopts a methodology which preserves some better chromosomes for next generations without applying any genetic algorithm operations [11].

In addition to this, if your problem definition contains so many criteria and rules, then your fitness function will be complicated and heavy-to-compute for each gene. Therefore, each gene from chromosome will have to be computed with this complex function. This will cause also an overhead. Here we will try to decrease this overhead; at the same time, we will try to handle the quality of chromosomes with football league approach.

A. Algorithm Proposal

1) Concept

The main concept is to think how a football league, teams, matches, players, transfers operate. In this paper, we will try to fit our genetic algorithm objects into football league objects and try to write scenario in terms of this concept. This will make us to have a better understanding and create our environment. Pointing to the previous paper [1] again, we assume our chromosomes as teams, genes as individual players, sub-populations as groups, crossovers as transfers, mutations as player performance modifiers. First of all, just as any other genetic algorithms do, we create a random population containing X teams (chromosomes); let's say X is **40.** These teams will be subject to group elections similar to football champions' league group elections. Each group contains Y teams; let's say Y is **4.** There will be two different levels, where each level has its own groups. The first level is "Primary league (PL)" where the better teams (chromosomes) take place, and the second level is "Qualification League (QL)" where the worse teams take place. So, if both levels include same number of groups and teams, with the given quantities above, we will have 5 groups containing 4 teams in both the "Primary League" and the "Qualification League". A "Primary League" is actually going to be the eye-pupil of the audiences (us) where there are some champions that should be avoided from brute cross over or mutations (elitist approach [11]). Some champion-candidates, which are the potentially good chromosomes where we will have them mutate or crossed-over with a lower percentage of occurring (the ones who take the second place after the champion). The third one will have a reasonable amount of crossovers or mutations a little higher than the second team. The last team in the group degraded to the lower league in the end of the iteration. A "Qualification League" is actually a training league, where some qualitied chromosomes appear and potentially will reach to championship. In this league, in the next iteration, any chromosomes have been exposed to extremely high cross-overs or mutations to create brand new chromosomes by disposing the non-qualitied chromosomes. If any new chromosome takes the first place, it will qualify for the "Primary league" in the next iteration. Initially (for the first iteration), random matches occur to determine which team goes to PL or QL. Until now, we have skipped how the chromosome evaluation occurs (matches played) to make a better understanding of the system concept. In the following, we

describe how league rules are implemented to show the vital part of this algorithm proposal.

2) Implementation

A simple football game means a match for two teams corresponding to having a match between two chromosomes in this approach. As in the introduction section, two chromosomes can play all genes like roulette selection or rank selection methodologies. Here we propose a new game theoretical approach to avoid this. To clarify what we are trying to say, we will try to adapt chromosomes to football game and load an intelligence to make them understand when to decide if the opponent is strong or weak, so that weaker side will withdraw from the match in a specific part of it. Because in the groups there will be some matches between two chromosomes.

As illustrated in Table I, minimal cost is targeted, and for the first match, team 1 is selected as winner in the middle of the entire chromosome process. For the second match, team 3 is selected as winner after 70% of genes have processed. The portion of matching genes is determined according to a decision mechanism based on game theory as described soon.

TABLE I. CHROMOSOMES WITH 10 GENES IN A GROUP HAVING MATCHES WITH RANDOM ORDERS

GROUP 1 – PRIMARY LEAGUE				
Match No	**Team No**	**Team Chromosome**	**Partial Cost**	**Energy Saved**
1	1	**1**-2-**3**-**4**-**5**-6-7-8-**9**-0	15	50%
1	2	**2**-1-**4**-**3**-**6**-5-8-9-**7**-0	85	50%
2	3	1-**3**-**5**-**6**-**7**-8-**9**-0-**2**-**4**	53	30%
2	4	0-**9**-**8**-**7**-**6**-5-**4**-3-**2**-**1**	75	30%

Assuming two chromosomes will have a match, one of them is weak and other one is strong. They do not know each other precisely, but they have a common knowledge on the probability of being strong or weak, which is calculated by using results of the previous matches. They will match their genes one by one actually in a **random order**, and they have two possible actions: Match (M) or Withdraw (W). If both decide for Match, a match occurs with that genes and fitness value computed. If one of them prefers Withdraw, and other one Match; then the game halts. Withdrawal side is counted as loser and the one who prefers to Match is counted as winner. Actually we do not want both of them to prefer withdraw, because if it happens we will miss to learn who is stronger and has the potential to beat the other, also we would waste our time for the matches occurred until then.

What about the psychology or intelligence of the teams? We can clarify this by instantiating it from real life. For the same scenario, if someone thinks himself stronger than the opponent in a specific time of a match, then he would always prefer to keep fighting. But, if he feels any weakness in a specific time of match, he would stop fighting and retreat in sometime. Because if he keeps fighting, he would lose in the end and also keeps losing much effort. So the earlier stopping loss, the better it is.

C1 \ C2	Match	Withdraw
Match	***i*** , *-i*	***a-i*** , *i*
Withdraw	***-a-i*** , *a-i*	-∞, -∞
Chromosome 2 is Weak (1-p)		

Fig. 1. Payoff table where Chromosome 2 is weak with (1-*p*) probability

C1 \ C2	Match	Withdraw
Match	***-i*** , *i*	***a-i*** , *-a-i*
Withdraw	***i*** , *a-i*	-∞, -∞
Chromosome 2 is Strong (p)		

Fig. 2. Payoff table where Chromosome 2 is strong with (*p*) probability

The variables in the game are described in Fig. 3.

c: stands for chromosome length

i: stands for number of matches played iteratively in a chromosome

a: stands for the award if a chromosome wins the game where the other one withdraws

$i \leq c$: *i* should be less than or equal to *c*.

Fig. 3. Variables of the game

Now let us describe the rationale behind the formulated game illustrated in Fig. 1 and Fig. 2. Assume Chromosome 1 is you and you have to decide, and Chromosome 2 is weak as in the Fig. 1. If both decides to match (MM) chromosome 1 will get a payoff '*i*', where chromosome 2 will get '–*i*'. So, as long as the number of played matches gets higher, and if you are strong, then you are boosted to keep going more with *i*. But if you are weak, you will get more penalty if you keep decide to be beaten by the stronger one. If the action profile is (MW), i.e. stronger one will decide to match, weaker one will withdraw, stronger will be awarded with '*a*' but will consume effort with '*i*', so payoff will be *a-i*. For the weaker one, his profit is lower at the beginning. As long as the number of played matches increases, and he is still the weaker one then he is boosted to leave the game increasingly. To be clearer, if you think yourself weak you wouldn't leave a match at the beginning, because you have still too many chances to reverse it. But, if there is little left to the end, and still you are weak, then it is easy to decide for withdrawal. This is actually what we try to provide by awarding the withdrawing side with '*i*'. This will encourage you to match and not to chicken out quickly. For the action profile (WM), you are assumed to be strong but still withdrawing, then you will lose an '*a*' award which is a guaranteed award but you give it up. Therefore, this is actually a lost. If you give up what you can get, it would be a loss. In addition, you also gave some effort with '*i*'. For the action profile (WW), this is actually a profile that we strictly don't want to happen. So, we will punish both sides with minus infinity. Therefore, we will try to keep them away from this choice.

Chromosome 2 is weaker than chromosome 1 with probability p. On the other hand, chromosome 2 is stronger with probability 1–p, and in that case it is going to be inverse of the decisions as shown in Fig. 2. So every one-shot game is actually a Bayesian game [10]. In every match, the payoff values change according to the number of matches played so far and probability of weakness (which is also variable according to the results of the previous matches). Hence, decisions may change at each match. The game ends if any of the players withdraw. So, on the contrary of our previous paper, this will avoid us of stuck with a single parameter for halting the game.

As we told, the modeled game consists of iterative play of Bayesian games, each with a different payoff matrix, according to the values of i and p. Since both players do not have a precise knowledge about who is stronger, they will have an expectation on the utilities when they choose action M or W. Expectations are told to be "ex-ante" in game theory language [10], since they do not know anyone's type exactly. Ex-ante expected utilities with respect to pure strategies are illustrated in Fig. 4.

C1 \ C2	**Match**	**Withdraw**
Match	***i-2pi, -i+2pi***	***a-i, i-2ip-ap***
Withdraw	***-a-i+ap+2ip, a-i***	**-∞, -∞**

Fig. 4. Bayesian Expected Payoff Table

When we analyze the expected payoff table illustrated in Fig. 4, we observe the following. Chromosome 1 continues to match, if the following condition hold:

$$i - 2pi \; > -a - i + ap + 2ip$$

On the other hand, chromosome 2 continues to match if the following condition hold:

$$-i + 2pi \; > -i - 2ip - ap$$

Therefore the Bayesian Nash Equilibrium (BNE) of the game is MM if:

$$\frac{2i}{a+4i} < p < \frac{a+2i}{a+4i}$$

The game continues with the next iteration only if p is in this range. Otherwise if $\frac{2i}{a+4i} < p$, then WM is the BNE and if $p < \frac{a+2i}{a+4i}$, then MW is the BNE.

In the proposed game model, there are several parameters such as the winning award a and the probability p that should be adjusted properly. Let us first talk about parameter a. If the chromosome length is 100, the value of i is between 1 and 100. If we set the award a to be a very small and constant value such as a = 1, both chromosomes continue to match (MM) if p is in the intervals given in Table II.

From Table II, it can be clearly seen that, in the very beginning of the game, sides will not withdraw and will match unless they do sense any strength from the opponent with a 60% or higher possibility. In other words, if it is weaker than its opponent with 60% probability or high, it will withdraw. Similarly, if it is weaker with 40% probability or low, its opponent will withdraw. But in the middle of the game, that interval quickly narrows down. If one side is a little bit stronger than the other, it will make the weaker one to withdraw quickly. Actually, it will be a prematurely end of a game. They do not pay any effort for a little award. They are tended to leave the game in the beginning of the game. This will not be a thing that we want.

TABLE II. CONTINUATION CONDITIONS FOR LOW WINNING AWARD

a = 1 (award is too low)	
i = 1 (Beginning of the game)	$\mathbf{0.4 < p < 0.6}$
i = 50 (Mid - game)	$\mathbf{0.4975 < p < 0.5025}$
i = 100 (End of the game)	$\mathbf{0.4987 < p < 0.5013}$

On the other hand, if we keep the award too high as in Table III, chromosomes will be too greedy to get this award, and play the matches nearly to the end unless they sense more than %98-99 strength rate of their opponent. Actually, this will converge to other selection methods which always evaluate all the genes of the chromosomes. So, keeping award too low or too high comparing to chromosome number would affect the results and may cause a disruption. Therefore, keeping nearly equivalent award to chromosome length would be a reasonable idea.

TABLE III. CONTINUATION CONDITIONS FOR HIGH WINNING AWARD

a = 10000 (award is too high)	
i = 1 (Beginning of the game)	$\mathbf{0.0002 < p < 0.9998}$
i = 50 (Mid - game)	$\mathbf{0.0098 < p < 0.9902}$
i = 100 (End of the game)	$\mathbf{0.019 < p < 0.981}$

Table IV illustrates the intervals when award is equal to the chromosome number. Here in the beginning of the game, both sides are not tended to withdraw. As long as the game resumes, the margins for withdrawing probability narrows reasonably.

TABLE IV. CONTINUATION CONDITIONS IF AWARD IS EQUIVALENT TO CHROMOSOME NUMBER

$a = c$ (award is equivalent to chromosome number)	
i = 1 (Beginning of the game)	$\mathbf{0.02 < p < 0.98}$
i = 30	$\mathbf{0.273 < p < 0.727}$
i = 50 (Mid - game)	$\mathbf{0.333 < p < 0.667}$
i = 100 (End of the game)	$\mathbf{0.4 < p < 0.6}$

Defining the weakness or strength probabilities (p) is another problem to resolve. Although there might be lots of approaches to define the value of p, we used a simple but also robust method. First of all, we increase a constant value of percentage for the winners probability where we decrease same amount for the losers in the same way. To make it clear, in the very beginning bets are fifty-fifty. When a gene wins a match, it gains percentage **(0.5 / *c*)**. This formula extracted from the test results as best practice until now. If the chromosome length is 100, probability of strength will increase by 0.5% for each match it wins, and decrease the same amount for each match it loses. So, if it loses all the matches, at the end of the game probability hits to 100%. But this percentage amounts are open to further investigation and

can vary a little bit from case to case. Another percentage change occurs from the difference of the cost of the individual match. For example, if cost values of two matching genes are 1 and 3, this is not same as another case where matching genes have cost values of 2 and 35. Cost values in 1 – 3 score are close to each other and it might not affect to the total result by itself as much as 2 – 35 does. So, we need to reflect this situation to the results in addition to constant value we have just mentioned. In each iteration in a group including PL and QL of that group, difference between the best cost and the worst cost is stored as a reference value to figure out margin of the cost values. In some problems, little values may refer to a huge cost difference, while in other problems a high value can stand for just a little difference. It actually depends on the problem definition, fitness function, etc. At each iteration, we find the reference value, and we the averages of these reference values. For example, if margin between the worst and best for iteration 1 is 80, and 70 for iteration 2, then our reference value is (80+70)/2 = 75 for the next iteration (i.e. iteration #3). And if a match in iteration 3 ends with 2 – 5 score (i.e. difference is 3), for this match winner gets 3/75 => 4% additional points besides 0.5% constant gain. We take the average of these two gains. As a result it gains 2.25% percentage, where the other loses the same amount. Although this approach yields reasonable results, finding the best method for updating *p* values deserve further investigation.

3) Selection Simulation

In this title, we will simulate how selection algorithm iterates including genetic algorithm methodologies. A brief introduction and explanation already given in the first title.

Firstly we consider a problem, such that there are 10 warehouses, and all warehouses installed with RFID readers to track products in the warehouse and know their identity and counts. This information is collected in a center. So, thanks to IoT, products can be monitored easily. There are 10 orders from different locations and we will try to satisfy this demand from the cheapest way as possible. There are some parameters which affect the cost like order-warehouse distance, warehouse product capacity etc. Here our chromosome will match with the orders respectively. The chromosome means which order will be provided by which warehouse. (i.e. 1-1-3-2 …, this means, the first and second order will be provided from warehouse 1, the third order from warehouse 3 and the fourth order from warehouse 2 and so on so forth.)

Given a population number with 4-teams group, in the 0-iteration, random chromosomes (teams) are created. All of them assigned randomly to their related groups. A group has two levels of league (e.g. Primary League and Qualification League). Separate groups have separate Primary Leagues and Qualification Leagues on behalf of theirs. For better understanding, we instantiate a group named ‘A’, and examine each iteration.

In Fig. 5, for each league, the first two teams play a match and the last two teams also play another match. (Note that a match between chromosomes consists of several matches between their genes.) The winners of each match play another match and losers will be determined from the winners’ final result. If a loser is the first team’s loser, then it will be the third team where the other one is the last. In the winners match, we always keep each gene cost in order to avoid recalculations in the forthcoming matches. Here we can spot the first and second teams in each league. The team 1 and 2 plays a match (with a=c) and team 2 beats team 1 in the 5th match (when i=5) due to probability of being weak is 70% for team 1 which exceeds 66,6%, the decision boundary of withdrawal. After the 5th match, team 2 and team 1 have the costs of 10 and 33, respectively. Similar match is played between team 3 and team 4. After the 7th match (i=7), team 4 beats team 3 with the costs of 20 and 55, respectively. Now the winner team of the first match (team 2) and second winner (team 4) will have another match. This match ends in the 8th match (i=8) for team 2 and team 4 with the costs 14 and 29, respectively. In the similar way, matches are played in the qualification league. This is the Alpha step of iteration 1 represented in the Fig. 6.

PRIMARY LEAGUE – GROUP A	
Team No	**Team Chromosome**
1	**1-3-3-4-5-9-8-7-6-2**
2	**2-1-4-3-6-5-8-9-7-0**
3	1-1-1-3-3-5-6-8-9-0
4	0-0-9-6-5-1-2-3-3-2

QUALIFICATION LEAGUE – GROUP A	
Team No	**Team Chromosome**
5	**4-8-4-0-1-2-3-2-1-1**
6	**5-7-3-8-9-9-9-6-0-3**
7	1-3-5-6-7-8-9-0-2-4
8	2-2-2-2-2-2-2-2-2-3

Fig. 5. Primary and qualification league of group A in the 0-iteration at the very beginning

PRIMARY LEAGUE – GROUP A		
Team No	**Team Chromosome**	**Latest Cost**
2	2-1-4-3-6-5-8-9-7-0	14
4	0-0-9-6-5-1-2-3-3-2	29
1	1-3-3-4-5-9-8-7-6-2	33
3	1-1-1-3-3-5-6-8-9-0	55

QUALIFICATION LEAGUE – GROUP A		
Team No	**Team Chromosome**	**Latest Cost**
7	1-3-5-6-7-8-9-0-2-4	13
6	5-7-3-8-9-9-9-6-0-3	27
8	2-2-2-2-2-2-2-2-2-3	44
5	4-8-4-0-1-2-3-2-1-1	54

Fig. 6. Alpha step of stage 1 for each league

In Alpha step, other than the mutation or crossover, a 'transfer' approach is implemented similar to the transfers in football world. Good players in some teams are transferred with a price or change of other players to boost up the team's power. Here we adopt this approach in a similar way. Every team encountered will have transfer within each other. While the matches between two teams played, winning genes are marked for each chromosome. At the end of the matches, winning individual genes are transferred to (copied to) the winner team (of course if it is feasible).

Fig. 7 illustrates the transfer approach. Underlined genes are ready to transfer to the winner team. This will make our winner chromosomes more powerful and fortify their weak points by getting rid of them and replacing the better ones from another team. This transfer process should be adopted for different problems appropriately. For example, if it is a permutation chromosome (all genes are different) [5], transfer process changes to appropriate one for this case. In every iteration, we keep the average reference value as we mentioned in the previous subsection. If we divide this reference value to chromosome length, then this will reflect the reference value for individual gene. After the match of individual genes, some of the genes in the loser team may be better than their counterparts in the winner team. In that case, if the difference is more than this individual reference point, then a transfer begins. But the transfer should be done in such a way that the resulting chromosome should be a permutation chromosome either. If we just replace two genes, then the replaced gene will exist twice in the chromosome. Thus, we find the index of the other occurrence of that gene and we compare it again with the loser chromosome's gene in the same index. If cost difference between these two genes are higher (where loser chromosome has the good gene again), then transfer occurs and the genes are replaced. If two genes have not already calculated yet, then skip transfer.

An example is illustrated in Fig. 8. Let us say that the individual reference value is 10, and when we match second genes of team 5 (which is the winner) and team 6 (the loser), the gene of team 6 (with a value of 7) is better with a difference higher than the reference value, say 11. So, a transfer operation is attempted. However, as it is a permutational chromosome, genes should not repeat. So we need to find the existing gene with value of 7 (Index:7) in the strong chromosome and check whether it is weaker than its counterpart in the loser chromosome. If it is not weaker, then no transfer happens. If there were no calculations on these index due to withdrawal, then again no transfer happens. A transfer happens only if existing gene with value of 7 has a higher cost than its counterpart in weak chromosome and the difference is higher than 11.

Moreover, weak team will be still same, so, weak teams can sometimes come up to a powerful team after some genetic algorithm operations. While we are keeping the strong teams which is elitism, we also create some weak teams in order to avoid convergence to local optima, which is a general problem in genetic algorithm [12]. Transfer operation depends on a parameter with a percentage. If it is zero, then no transfer happens, if it is 100, then all winner genes transferred to winner chromosome, if it is 50, half of the winner genes transferred and so on.

Before Transfer	
Team No	**Team Chromosome**
5	4-8-4-0-1-2-3-2-1-1
6	5-7-3-8-9-9-9-6-0-3
After Transfer	
Team No	**Team Chromosome**
5	4-8-4-0-1-2-3-2-1-1
6	5-8-3-8-1-9-9-2-0-3

Fig. 7. A transfer operation between winner and loser team where team 5 is the loser

Before Transfer		
Team	**Team Chromosome**	**Brief Explanation**
5	1-2-3-4-5-6-7-8-9-0	Team 5 is the winner. Cost difference between 2nd genes is -11 (gene of team 6 is better). Find (7) in team 5. Compare with the gene of team 6 in same index. If again gene of team 6 is better and difference is more than 11, do transfer and replace (7) with (2) in team 5.
6	5-7-2-1-4-0-9-8-3-6	
After Transfer		
Team No	**Team Chromosome**	
5	1-7-3-4-5-6-2-8-9-0	
6	5-7-2-1-4-0-9-8-3-6	

Fig. 8. A transfer operation for permutational chromosome between winner and loser team where team 5 is the winner and team 6 is the loser.

PRIMARY LEAGUE – GROUP A		
Team No	**Team Chromosome**	**Operation**
2	2-1-3-3-6-9-8-7-7-0	No operation
1	1-3-3-4-5-9-8-7-6-2	Crossover with below
4	1-1-9-6-5-1-2-3-3-2	Crossover with above
3	1-1-1-3-3-5-6-8-9-0	Playoff with QL-1st

QUALIFICATION LEAGUE – GROUP A		
Team No	**Team Chromosome**	**Operation**
7	2-3-5-6-7-2-9-0-2-4	Playoff with PL-4th
6	5-8-3-8-1-9-9-2-0-3	Mutations and Crossover with below
5	4-8-4-0-1-2-3-2-1-1	Mutations and Crossover with above
8	2-2-2-2-2-2-2-2-2-3	Drop it, generate new

Fig. 9. Beta step of iteration 1 for each league

For the Beta step, we will have a playoff match between the last team of PL, and the first team of QL. This will give a chance to QL teams for heading the championship. If PL team beats, then ranking stays same. If QL team beats, then PL team degraded to QL, and QL team will rise to PL. Transfer operation also occurs in this match. Moreover, in the Beta step, genetic algorithm operations are applied. Fig. 9 shows which operations will be applied to which team and how. Last team of QL vanishes and a new random team appears for the next iteration.

After Alpha and Beta steps, the teams are ready to other iteration where all operations are identical to the previous iteration. After a specified number of iterations, the game ends. Alternatively, game may end after running for a specified time or any other rule can be determined for termination decision. If the game ends, a final stage occurs. This corresponds to champions league in the football world. All the first ranked teams pushed into another league and their full costs calculated without Bet Prediction Selection operations. It is simply a ranking operation. The fittest chromosome will be our solution chromosome. If we have 6 groups, then 6 different PL exist. This means that 6 of the first team will be selected for champions league.

The rationale behind the operations given in Fig. 9 can be described as follows: For PL, keeping the first team as-is means an elitist approach. Crossing over the second and the third by not mutating or mutating with a small amount keeps good chromosomes in a well-condition. They took the second and the third place in PL meaning we can't neglect their performance, and their crossover pair operation may create better teams. Playoff between PL and QL provides us to discover good chromosomes from QL. This playoff operation gives a great chance to them to make the things better. The second and the third in QL will change themselves too much with crossovers and lots of mutations. This will give a chance to evolve to something better. For the last team, no chance! Just take over your place to another random team, you are not valuable. Another random team can make their way to the champion's way better than you. Qualification league also prevents Primary League to stuck local optima as we value low quality chromosomes as well.

III. TEST RESULTS

We test the proposed algorithm in an NP-Hard generalized assignment problem in IoT domain. But firstly, for the sake of clarity and simple understanding, we consider a simple problem with 10 different warehouses located on a city and there are 10 orders to welcome from these warehouses. Again just for easier understanding, we keep a simple fitness function (i.e. sum of [order number * warehouse number] – warehouse number). The objective is to minimize this fitness function. Number of orders (and also number of warehouses) is the chromosome length. So, without genetic algorithm, we can easily figure out that the best chromosome is the chromosome which is ordered reversely from 9 to 0 (9-8-...-2-1-0) to keep fitness function minimum. We run each of the bet prediction selection algorithm and elite-roulette-rank mixed selection algorithm ten times and take their average. Elite, roulette and rank selection algorithms performed sometimes better than each other, or worse, but in overall they yield very similar and indistinguishable results, that is why we averaged them and run them equal-times. For all algorithms, population size is 32. Results are illustrated in Table V.

Mixed selection seems to be faster for short chromosome length. Because Bet Prediction Selection (BPS) has some operational loads and it cannot show its power for such short and easy calculations. Nevertheless, the obtained average cost values are close to each other. BPS algorithm averagely played almost 8 matches of 10, which means 2 matches (genes) are saved. This is shown in Table V as average energy saved.

TABLE V. CONSOLIDATED TEST RESULTS FOR SHORT CHROMOSOME LENGTH FOR 10, 100 AND 1000 ITERATIONS (B: BET PREDICTION SELECTION, M: ELITE/RANK/ROULETTE MIXED SELECTION)

Algo.	Iter.	Best Chromosome Found	Avg. Cost	Avg. Time (msec)	Avg. Energy Saved
B	10	9 8 6 7 4 2 5 3 0 1	146	103	21.1%
M	10	9 8 4 5 6 7 3 1 0 2	140	55	-
B	100	9 8 7 6 5 2 4 3 0 1	129	701	22.8%
M	100	8 9 5 4 6 7 3 2 1 0	130	149	-
B	1000	9 8 7 6 5 4 2 3 0 1	127	5124	20.5%
M	1000	9 8 7 6 5 4 3 0 1 2	125	1235	-

Let us evaluate how BPS energy-save operation affects the execution time. For this purpose we simulate a long time taking fitness function with artificial delays. Table VI illustrates the results, where the first row corresponds to BPS algorithm that evaluates the whole chromosomes, and the second row corresponds to the BPS algorithm with energy-save operation as described in the previous section (a portion of genes match with each other, rather than all of them). The obtained results suggest that the proposed BPS algorithm saves significant time, while there is just a negligible cost difference compared to the BPS algorithm that evaluates the whole chromosomes.

TABLE VI. PERFORMANCE COMPARISON OF BPS ALGORITHM WITH AND WITHOUT ENERGY-SAVE FOR A FITNESS FUNCTION WITH HIGH TIME-COMPLEXITY

Algo.	Length	Iter	Avg. Cost	Has Energy Save	Saved Energy	Avg. Secs
B	100	10	~210910	No	-	75
B	100	10	~211100	Yes	%20.9	62

Now, we compare Bet Prediction Selection and mixed selection algorithms (Elite/Roulette/Rank) for a fitness function with high time complexity (such as a chromosome's cost can be calculated within nearly 1 seconds). We set the chromosome length to 100 and keep the other features the same. The results are shown in Table VII.

TABLE VII. TEST RESULTS FOR COMPLEX FITNESS FUNCTION FOR 5 AND 10 ITERATIONS, 100 CHROMOSOMES (B: BET PREDICTION SELECTION, M: ELITE/RANK/ROULETTE MIXED SELECTION)

Algorithm	Iteration	Avg. Cost	Avg. Time (sec)	Avg. Energy Saved
B	5	~218000	119	23.4%
M	5	~230000	131	-
B	10	~211000	208	21.8%
M	10	~221000	232	-

Test results illustrated in Table VII shows that for long time taking fitness functions, Bet Prediction Selection is better than Mixed selection in terms of both the obtained cost and time. This indicates the contribution of the proposed game-theory based energy-saving operation.

For a real use case instead, we have simulated some courier and order information from a car manufacturing sector. Maintenance and repairing centers request some spare parts of car such as rear mirror, side mirror, door handles, windshield wiper etc. These centers use these pieces for the cars for their customers who need a repair. All spare parts are tracked by the system with RFID technology as a stock management. Couriers which carry these parts from warehouse to centers are tracked by GPS. Each warehouse has one courier for working on behalf of them. One courier can carry one order at a time. If courier is assigned 2 orders, then he needs to deliver the first order and return to warehouse and collect the next order. Orders come from centers due to customer demands. Any part does not reside in any warehouse. Some warehouses have light parts (like rear mirrors etc.) where other warehouses have heavy parts (like mechanic parts, tire assembling parts etc.). Weights are measured with weight-meter sensors. Orders should be collected with the shortest way with less traffic from the warehouses. They are also monitoring the traffic status of the city for any jams or closed roads, etc. There are four warehouses and one factory. The factory contains all the products but it is further away from the orders than warehouses. So, any order can be provided from factory but it increases the cost because of the distance. Moreover, we want our couriers to be assigned with the same amount of orders to satisfy fairness and avoid courier overload. This means they have many inputs to evaluate. We have collected a basic problem which meets this requirement and generated some sample data compatible with the data from their IoT backbone and created a fitness function which values the distances between transporters, warehouses and order centers. If any product does not exist in warehouse and order is attempted to collect from there, the **penalty** is 10M. Because that would not be a feasible solution. Traffic condition rate is ranged from 1 to 7, where 7 means very busy roads. Factory is further away from the orders at least 20km to 40km for return trip. Orders are in a circular range such that the maximum distance is 20 km to a warehouse for return trip. Orders contain products. Products have their own weights (1, 2, 3, 5, 30, 50kg). Fitness function is the sum of (*Product Weight * Distance * Traffic Rate*). The problem to minimize the fitness function is a sort of multi-task generalized assignment problem [2] which is proven to be NP-Hard. We run the same algorithms for this problem (we take the average of 10 runs) with 100 orders and the results are given in Table VIII.

TABLE VIII. PERFORMANCE TEST RESULTS FOR REAL USE CASE WITH 100 ORDERS (B: BET PREDICTION SELECTION, M:MIXED SELECTION, NFS: NO FEASIBLE SOLUTION)

Algorithm	Iteration	Avg. Cost	Avg. Time (sec)	Avg. Energy Saved
B	50	68k	3	35%
M	50	NFS	2	-
B	100	67k	6	34.1%
M	100	NFS	4	-
M	300	NFS	13	-
B	500	61k	30	31.5%
M	500	78k+NFS	20	-

From Table VIII, the performance contribution of the proposed BPS algorithm can be observed evidently. As fitness function is easy to compute, execution times of BPS algorithm is a bit more than the other selection algorithms, but BPS can reach to a feasible solution within very short time in a less iteration number, where other algorithms start to find some feasible solutions after 500th iteration where some of their solutions are still not feasible.

Another test is using same environment just only mimicking fitness function to be long-computed by using a delay. (A chromosome is calculated within nearly 0.5 sec). For this case, BPS is better in terms of both time and cost as shown in Table IX.

TABLE IX. PERFORMANCE TEST RESULTS FOR REAL USE CASE WITH 100 ORDERS AND DELAYED FITNESS FUNCTION (B: BET PREDICTION SELECTION, M:MIXED SELECTION, NSF: NO FEASIBLE SOLUTION)

Algorithm	Iteration	Avg. Cost	Avg. Time (sec)	Avg. Energy Saved
B	50	70k	340	%34
M	50	NFS	365	-

IV. CONCLUSION

In this study, we developed a novel approach for genetic algorithm selection methods that benefits from analogy of football leagues and uses the idea of predicting the result after iterating only a portion of genes, hence reducing the execution time. The amount of these portions are decided according to a novel concept based on Bayesian game theory. The proposed algorithmic idea is especially suitable for many problems in IoT domain where the time is valuable. This paper describes the proposed method in detail and illustrates its benefits through implementation on an NP-hard problem in industry with various system settings.

Although the proposed algorithm yields good results, it is still open to further development. There are several operations that should be adjusted to find the best practice, such as league creation approach and prediction of strength of a chromosome. Moreover, more extensive comparative study is required to reveal its performance contribution over other existing selection algorithms in the literature.

ACKNOWLEDGMENT

This work is supported by Koç Digital R&D Center.

REFERENCES

[1] K. Aytaç, Ö. Korçak, "IoT Based Smart Staff Allocator in Quick Service Restaurants", *in Proceedings of the 23rd FRUCT*, 2018.

[2] L. Özbakir, A. Baykasoğlu, P. Tapkan, "Bees algorithm for generalized assignment problem", *Applied Mathematics and Computation*, vol. 215, 2010, pp. 3782–3795.

[3] Wikipedia official website, Facility Location Problem, Web: https://en.wikipedia.org/wiki/Facility_location_problem [Online][Accessed 30- Jul- 2019]

[4] R J. Aumann, "What Is Game Theory Trying to Accomplish?", Frontiers of Economics, Edited by K. Arrow and S. Honkapohja, 1985 pp. 28-76.

[5] G. Ucoluk. “Genetic Algorithm Solution of the TSP Avoiding Special Crossover and Mutation”, *Intelligent Automation and Soft Computing*, vol.3, 2002.
[6] N. Kaur and S. K. Sood, “A Game Theoretic Approach for an IoT-BasedAutomated Employee Performance Evaluation”, IEEE Systems Journal, Vol. 11, No. 3, 2017. pp. 1385
[7] E. Abd-Elrahman ; H. Afifi ; L. Atzori ; M. Hadji ; V. Pilloni, “IoT-D2D task allocation: An award-driven game theory approach”, *23rd International Conference on Telecommunications (ICT)*, 2016.
[8] Wikipedia official website, Fitness proportionate selection, Web: https://en.wikipedia.org/wiki/Fitness_proportionate_selection [Online][Accessed 31- Jul- 2019]
[9] J. Khalid, “Selection Methods for Genetic Algorithms”, *International Journal of Emerging Sciences*, vol. 3, 2013, pp. 333-344.
[10] Y.Shoham, K.Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*, Cambridge University Press, 2008.
[11] G.Soremekun, Z.Gürdal, T.Haftkac, L.T.Watson, “Composite laminate design optimization by genetic algorithm with generalized elitist selection”, *Computers & Structures*, vol.79, no. 2, 2001, pp. 131-143
[12] M.Rocha, J.Neves, “Preventing Premature Convergence to Local Optima in Genetic Algorithms via Random Offspring Generation”, In: Imam I., Kodratoff Y., El-Dessouki A., Ali M. (eds) Multiple Approaches to Intelligent Systems. Lecture Notes in Computer Science, vol 1611.(1999) Springer, Berlin, Heidelberg.
[13] G Cornuéjols, G Nemhauser, L Wolsey, “The Uncapicitated Facility Location Problem”, *Cornell University Operations Research and Industrial Engineering*, 1983.