IJCSI International Journal of Computer Science Issues, Volume 14, Issue 4, July 2017
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org https://doi.org/10.20943/01201704.3543

A Self-Adaptive Multi-Resources Suppliers System

Ihab Sbeity, Mohammad-Baker Elkalla and Mohamed Dbouk

Computer Science Department, Faculty of Sciences,
Lebanese University, LEBANON

Abstract

Dealing with a suppliers/demands system with functional transportation cost is usually considered as a classical transportation problem and can be solved with normal linear programming processes, but what adds to it, when the cost in the supply points is to be taken into consideration, demands at each demand point varies with time, and complexity of the whole system grows. Furthermore, introducing a constant cost to services and/or transportations, which is the case in several real distributed systems, makes the model more sophisticated to be solved. In this case, additional measures prevent us from treating the model as a classical transportation problem; instead, we have to search another technique to grant satisfactory results. This paper proposes an approach that enables a system to auto-reconfigure itself at runtime whenever environmental variables vary, with the goal of minimizing costs while satisfying the demands, while taking into consideration the different type of costs (constant/functional, service/transportation, etc.). Furthermore, this work aims to build a framework to find the optimal configuration of services/transportations in a multi-resources suppliers system requiring an auto adaptation facing the variation of demands. Our proposed framework, called GATRA, is a combination of the Genetic Algorithms and the Transportation problem techniques.

Keywords: *Mutli-resources Systems, Auto-adaptive, Genetic Algorithm, Transportation Problem.*

1. Introduction

As distributed systems grow in size and complexity in response to economical needs, health and life demands, it is highly needed to integrate new algorithmic factors in these systems which allow them to conform all the constraints encountered through their lifetime and helps them in fine-tuning and optimal reconfiguration while operating continuously [13].

Many researchers have induced papers on solving reconfiguration problems in distributed systems. Some of which used methods like (the main criterion method (ε-constraint) that transforms the whole problem into a single objective problem [14], but a main weakness point appears for that summarized by altering the essence of the original technical problem and that's due to the existence of more than one index that must be considered throughout the optimization problem. Others [10] used aggregation functions for the same purpose (transform into a single objective problem), but the weakness aroused here was due to the incompatibility of the different criteria which leaded to the need to create global functions to convert criteria to the same measurement unit and thus taking more objective functions ending with a formulation problem.

Some researches covered reconfiguring the network in distributed systems [8], some, regardless of the problem formulation, covered searching for the solution [11], which uses the "brute force" which generates the entire space of the candidate solutions and then chooses the best one among them through many suggested heuristic [3,1], meta-heuristic [15,7,17] (Which have been very popular in the last decades), and multi-agent technologies [4]. Evolutionary algorithms (EAs) have been used extensively for solving multi-objective optimization problems in several areas [6, 16] and in particular obtaining the Pareto-optimal solutions for DSR problems [9].

As known, distributed systems have different topologies. In this work, the system we are interested in is a typical demand/supply zones system, where each supply zone is composed of multi-resources that provide services for the demands zones. Each supply zone is connected to all the demands zones via transportation channel. Examples of these systems can cover all sides of our real life like 'power distributed systems' which consists of zones of generators (supply zones) which are connected to several residence regions (demand zones). Similarly, we can talk about 'networking distributed systems' (routers, switches, etc), 'water supply systems' (water pumps, water containers.), 'Gas supply systems', and many other similar functional systems.

The optimal reconfiguration usually consists on minimizing the cost needed to satisfy the demands. Moreover, the multi-criteria optimization reconfiguration solution is based on evaluating certain indices that represent multiple purposes. In the restricted presence of only functional transportation costs, the optimal reconfiguration is obtained by solving a classical transportation problem with objective to minimize the transportation cost while satisfying the demands. However,

other meaningful constraints may be added to the system and that makes impossible to consider it as a classical transportation problem. Hypothesizes like (constraint A) "only one resource at each supply zone may be active", or (constraint B) "there is a service cost for each resource that may be equal to a constant value" are some examples of these constraints. Systems with only constraint B are called in this paper, GATRA systems.

In front of the above problematic, in this work, we propose an approach based on the combination of the genetic algorithm and the transportation problem techniques, to achieve the optimal reconfiguration of GATRA systems, to make it auto-adaptive facing the on-time variation of demands. Our model takes only into consideration (constraint B) described above, and it can be generalized to cover any other constraint. A genetic algorithm is a type of searching algorithm [2]. It searches a solution space for an optimal solution to a problem. The key characteristic of the genetic algorithm is how the potential solution is evaluated through a fitness function that presents a measure of how "good" the solution is. In special case where the service cost is constant, and the transportation cost is functional, our approach consists on embed the evaluation of transportation problem in the genetic algorithm fitness function.

Although, the combination of genetic algorithm with transportation problem has been the key solution of many research works, our approach is simpler and quite direct to be applied in the model described above. We list as example the work of [12], where authors have addressed the case of transportation problem with Piecewise non-linear cost function. The complexity of their solution is considerable and inefficient when applied to our case.

The rest of this paper is organized as follows. In section 2, we recall the basic concept of the transportation problem and the genetic algorithm. In Section 3, we formally describe our transportation system, and we discuss its possible extensions. In Section 4, we propose the solution to find the optimal configuration of the system, and that makes it auto-adaptive facing the variation in demands. As we mentioned before, our approach is based on the embedding of the transportation problem objective function in the fitness function of the genetic algorithm. For that reason we will name the approach GATRA (Genetic Algorithm with TRAnsportation objective embedded function), that will solve GATRA systems. In section 5, we present some numerical experimentation and analysis. Section 5 concludes our paper and describes our ongoing works.

2. Background

The purpose of this work is to find the optimal resources configuration facing the variation of demands. Our approach embeds the objective function of transportation problem into the fitness function of a genetic algorithm in order to make a GATRA system auto-adaptive. So, let us first recall the concept of transportation problem and genetic algorithm.

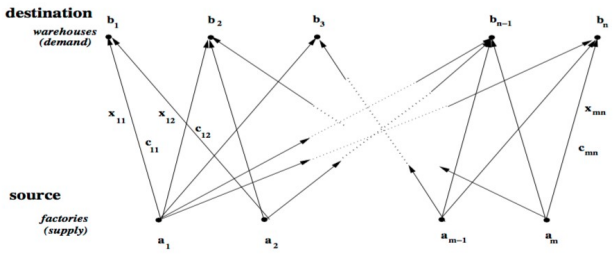
2.1 Transportation theory

From Wikipedia, the "Transportation Theory" is a name given to the study of optimal transportation and allocation of resources. The transportation problem is a special class of linear programming problem, which deals with shipping commodities from source to destinations. The objective of the transportation problem is to determine the shipping schedule that minimize that total shipping cost while satisfying supply and demand limits. The transportation problem has an application in industry, communication network, planning, scheduling transportation and allotment etc.

To apply such algorithm we shall be given:

- The level of supply at each source and the amount of demand at each destination.
- The unit transportation cost of the commodity from each source to each destination. Thus, the transportation cost is a linear function of the shipped quantity.

A destination can receive its demand from more than one source. The objective is to determine how much should be shipped from each source to each destination so as to minimize the total transportation cost. The following figure [5] summarizes the process.



The figure represents a transportation model with m sources and n destinations. A node represents each source or destination. An arc joining the two nodes represents the route between a source and destination. The amount of supply available at source i is a_i, and the demand required at destination j is b_j. The cost of transporting one unit from source i and destination j is c_{ij}.

Let x_{ij} denote the quantity transported from source i to destination j. The cost associated with this movement is cost×quantity = $c_{ij}x_{ij}$. The cost of transporting the commodity from source *i* to all destinations is given by:

$$\sum_{j=1}^{n} c_{ij}\, x_{ij} = c_{i1}x_{i1} + c_{i2}x_{i2} + \cdots + c_{in}x_{in}$$

Thus, the total cost of transporting the commodity from all the sources to all the destinations is:

Total Cost =

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} = c_{11}x_{11} + c_{12}x_{12} + \cdots + c_{1n}x_{1n} + c_{21}x_{21} + c_{22}x_{22} + \cdots c_{2n}x_{2n} + \ldots + c_{m1}x_{m1} + c_{m2}x_{m2} + \cdots c_{mn}x_{mn}$$

In order to minimize the transportation costs, the following problem must be solved:

$$\text{Minimize } z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\, x_{ij},$$

$$\text{subject to } \sum_{j=1}^{n} x_{ij} \le a_i \text{ for } i = 1, \ldots, m$$

$$\text{and } \sum_{i=1}^{m} x_{ij} \ge b_i \text{ for } j = 1, \ldots, n$$

$$\text{where } x_{ij} \ge 0 \text{ for all i and j.}$$

Where *z* is the objective function and the other are the problem constraints. The first constraint says that the sum of all shipments from a source cannot exceed the available supply. The second constraint specifies that the sum of all shipments to a destination must be at least as large as the demand.

The above implies that the total supply $\sum_{i=1}^{m} a_i$ is greater than or equal to the total demand $\sum_{j=1}^{n} b_j$

When the total supply is equal to the total demand $(\text{i. e.} \sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j)$ then the transportation model is said to be balanced. In a balanced transportation model, each of the constraints is an equation:

$$\sum_{j=1}^{n} x_{ij} = a_i \text{ for } i = 1, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} = b_j \text{ for } j = 1, \ldots, n$$

A transportation model in which the total supply and total demand are unequal is called unbalanced. It is always possible to balance an unbalanced transportation problem.

The solution of transportation problem consists on minimizing its objective function. Given a balanced transportation problem, it is easy to find its solution by applying well known algorithm such that Vogel algorithm [18]. The solution process will be embedded in our solution of GATRA model.

2.1 Genetic algorithms

Genetic Algorithms (GAs) are adaptive heuristic search algorithms used to solve optimization problems. They are based on the evolutionary ideas of natural selection and genetics and exploit historical information to direct the search into the region of better performance within the search space. Their basic techniques are based on simulating processes following the principle “survival of the fittest” by Charles Darwin [2].

GA’s proved that they do not break easily even if the inputs changed slightly, noise occurred, or dimensions increased. They have proved their efficiency in searching a large state-space, multi-model state-space, or n-dimensional surface, and may offer significant benefits over other optimization techniques (linear programming, heuristic, depth-first, breath-first, and praxis).

A genetic algorithm is based on choosing the fittest among individuals after passing several generations, every individual is a chromosome (related to chromosomes in our DNA) and represents a point in a search space and is a possible solution. And every generation is a group of these individuals (Chromosomes). The individuals in the population are then made to go through a process of evolution.

GAs are based on an analogy with the genetic structure and behavior of chromosomes within a population of individuals using the following foundations:

- Individuals in a population compete for resources and mates.
- Those individuals most successful in each 'competition' will produce more offspring than those individuals that perform poorly.
- Genes from `good' individuals propagate throughout the population so that two good parents will sometimes produce offspring that are better than either parent.

- Thus each successive generation will become more suited to their environment.

The search space of a GA is a population of maintained individuals, in which each of them represents a possible solution of the problem. We can describe an individual to be as a finite length vector of variables, in terms of some alphabet, usually the binary alphabets {0,1}. So now these individuals are like chromosomes and their variables are like the genes in the chromosome. Then a fitness value is assigned to each individual that represents its ability to 'compete'. Those individuals of the highest fitness values are chosen to generate the new generation 'off-springs' with better fitness values, and thus nearer to the optimal fitness score.

3. GATRA Systems

In this section, we describe GATRA systems model, and we represent the techniques used to solve it. We make a brief about the main model, define its variables, and then explain the cost-states where the GATRA technique can be applied. After that, we state the stages of our algorithm through the pseudo of each function used in it followed by a real example to illustrate the steps used in the pseudo.

3.1 Model definition

A general mutli-resources transportation system is composed of a number (n) of supplying sources (S), and number (k) of demand points (D) (figure 1). Each supplying point S_i contains m_i sub-sources (sub-suppliers) (s_{i1},s_{i2},…,s_{im_i}) with known capacities for each, and each supplying point is connected to all the demand points with c_{ij} being the transportation cost between S_i and D_j. For example, in a power provider system, the supplying sources may be zones of generators which are connected to several residence regions (demand zones). Each supplying zone may hold multiple generators that provide power to the connected demand zones. Of course, several scenarios related to the system functionality and the engendered cost may be considered. We discuss the different scenarios later on. Let us first, formulate our model.

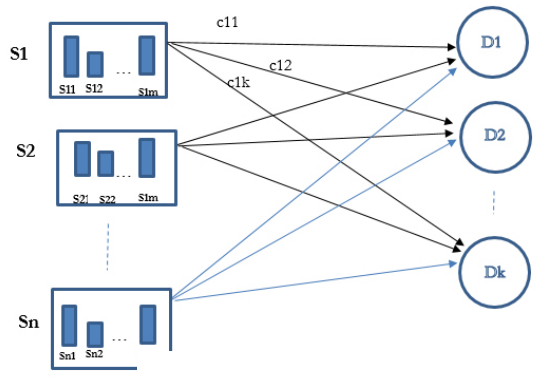


Fig 4.1

Formally, a multi-resources transportation system is defined by the following variables:

Let:

- m is the number of suppliers.
- n is the number of demands.
- S_i : The supplier's name where $i \in \{1,2, \dots, m\}$ where m is the number of suppliers.
- D_j : The demand's name where j$\in \{1,2, \dots, n\}$ where n is the number of demands.
- b_j : The demand's quantity at demand point D_j
- c_{ij} : The cost of transportation per unit from S_i to D_j. (**transportation cost**)
- m_i The number of sub-suppliers in each supply S_i.
- s_{ia} be the capacity that each sub-suppliers in S_i such that:
  - $i \in \{1, \dots, m\}$ Where m is the number of suppliers.
  - $a \in \{1,, \dots, m_i\}$ Where m_i is the number of sub-suppliers in each supply S_i.
- r_{ia}: The cost to turn on the s_{ia} sub-supply (**service cost**).
  - $i \in \{1, \dots, m\}$ Where n is the number of suppliers.
  - $a \in \{1,, \dots, m_i\}$ Where m_i is the number of sub-suppliers in each supply S_i.
- x_{ia}^j: The amount supplied from s_{ia} to D_j.
- x_{ij} : The amount supplied from S_I to D_j thus : $x_{ij} = \sum_{a=1}^{m_i} x_{ia}^j$
- m_{ia} : The indicator that indicates whether s_{ia} is turned on or of and will have the values 1 (source is on) or 0 (source is off).

The previous variables define a general model whose complexity increase with their type and their values. In this work, we only focus on the type of the costs r_{ia} and c_{ij}, although considering other possibilities is also crucial, but it will be addressed in future works, e.g. criteria on the number of sub-suppliers turned on at a time. In our study, the indicator m_{ia} does not have a meaning.

Fundamentally. There are two types of service/transportation costs that can be considered:

- Function of the supplied quantity (type F)
- Constant; independent of the supplied quantity (type C).

Again, the costs' type may be different at each supply points, however, for simplicity purpose, we consider that type of a cost has the same type at each supply point. Thus there are 4 scenarios of (service, transportation) costs' types: (F,F), (F,C), (C,F), and (C,C).

Notice that, in case of the (F,F) scenario, the model can be considered as a classical transportation model: each sub-supply s_{ia} is considered as independent supply point with a total transportation cost to the demand point D_j is equal to $r_{ia} + c_{ij}$. The LP model corresponding to the transportation model is given as:

$$\text{Minimize } z = \sum_{i=1}^{m}\sum_{a=1}^{m_i}\sum_{j=1}^{n}(c_{ij} + r_{ia})x_{ij},$$

$$\text{subject to}$$

$$\sum_{j=1}^{n} x_{ia}^{j} \leq s_{ia} \text{ for } i = 1, \dots, m$$

$$\sum_{i=1}^{m}\sum_{a=1}^{m_i} x_{ia}^{j} \geq b_j \text{ for } j = 1, \dots, n$$

$$\text{where } x_{ia}^{j} \geq 0 \text{ for all i, a and j.}$$

On the other hand, the resolution of the model in the three others scenarios, i.e. at least one cost function is constant, cannot be considered as a simple resolution of a transportation problem, and needs to introduce other technique to solve the model. A GATRA system is basically a transportation system having such kind of constant cost. In the next section, we show how to solve this kind of problems by introduction the genetic algorithm to the resolution process

3.2 Model Resolution

As it has been presented in the previous section, a GATRA is a kind of multi-resources transportation system with at least one of the service/transportation costs is constant. In this case, the system cannot be solved as a classical transportation problem, and needs intelligent algorithms to be explored.

To solve a GATRA system, we propose to use the genetic algorithm (GA), while embedding a transportation problem's objective function to the GA fitness function in case where, exclusively, one of the service/transportation cost is constant, i.e. scenarios (F,C) or (C,F). However, if the both costs are constant, a pure genetic algorithm is applied to solve the model (scenario (C,C)).

The following table summarizes the different scenarios rising from a multi-resources transportation system, and the techniques used in each scenario to solve the model.

(service cost, transport cost)	(F,F)	(F,C)	(C,F)	(F,F)
Techniques used	Pure TR	GA + embedded TR		Pure GA
		GATRA		

In the following, we focus on the resolution of GATRA system in case of (F,C) or (C,F) scenarios using the GA with embedded transportation problem function (called GATRA method), as the resolution of (C,C) is purely based on genetic algorithm and can be easily derivate from GATRA method.

Moreover, we will show the steps used with GATRA method basing on the following simple example, and then we will formally generalize the method.

Example:

Let us consider a case where we have two supplying points (2 resources) and two destinations (2 demands), each supplying point contains 2 sub-resources with the following capacities:

- resource S_1 : s_{11} = 35 ; s_{12} = 45
- resource S_2 : s_{21} = 105 ; s_{22} = 40

The 2 destinations (demands) values are: b_1=80; b_2=60.

In this example, we consider the case where the service cost is constant at each sub-resource (10% of its capacity), but the transportation cost is a function the supplied quantity (scenario (C,F)).
s_{11}= 4; s_{12}= 6; s_{21}=8; s_{22}=13; s_{31} =13; s_{32} =11; s_{41}=10; s_{42}=8

Recall that our objective is to find the combination of sub-resources that should be active (on service) in order to satisfy the demand while minimizing the total cost. Using GA, our method will systematically produce a set of combinations via the crossover/mutation techniques, and pass them to the fitness function that represent the total cost. The service cost is directly calculated, i.e. for each active sub-resource active in the combination, the method

sums 10% of its capacity. Once a combination of active sub-resources is given, the model under this combination is considered as a classical transportation problem by neglecting the service cost, and the objective function of the problem is then summed to the GA fitness function to obtain the total cost of the system.

Back to our example, in our GA, the chromosome is a matric of genes, where each gene will indicate the state of the (resource/related sub-resource) w.r.t a specific destination (demand). In our example of 2 resources/2 destinations, we do have 4 Genes that will form each chromosome. The rows of the matrix represent the demand, the columns represent the resources. Let us start by considering the two following chromosomes (randomly generated by the algorithm)

	S1	S2
D1	(0,0)	(0,0)
D2	(0,1)	(1,1)

(0,0)	(1,0)
(1,0)	(0,0)

Chromosome 1 Chromosome 2

For example, in chromosome 1, only the sub-resource 2 of the resource 1 is supplying the demand 1. However, the two sub-resources 1 and 2 of the resource 2 are supplying the demand 2. Similarly, in chromosome 2, sub-resource 1 of resource 1 is supplying D2, and sub-resource 1 of resource 2 is supplying D1.

After we defined the starting chromosomes, follows the crossover operation that will result in two new chromosomes added to the original parent's chromosomes to form a group of 4 chromosomes. There is several strategies that may be used to crossover two matrices. We have chosen the lower triangular crossover technique, which is based on keeping the lower triangular part of one parent chromosome and taking the upper remaining part from the other parent, as shown in the following:

$$\begin{matrix}(0,0) & \mathbf{(0,0)} \\ (0,1) & (1,1)\end{matrix} \text{ X } \begin{matrix}\mathbf{(0,0)} & (1,0) \\ \mathbf{(1,0)} & \mathbf{(0,0)}\end{matrix} = \begin{matrix}\mathbf{(0,0)} & \mathbf{(0,0)} \\ \mathbf{(1,0)} & \mathbf{(0,0)}\end{matrix}$$

$$\begin{matrix}(0,0) & (1,0) \\ (1,0) & (0,0)\end{matrix} \text{ X } \begin{matrix}\mathbf{(0,0)} & (0,0) \\ \mathbf{(0,1)} & \mathbf{(1,1)}\end{matrix} = \begin{matrix}\mathbf{(0,0)} & \mathbf{(1,0)} \\ \mathbf{(0,1)} & \mathbf{(1,1)}\end{matrix}$$

As a result we get these 4 chromosomes ready for mutation.

(0,0)	(0,0)
(0,1)	(1,1)

(0,0)	(1,0)
(1,0)	(0,0)

(0,0)	(1,0)
(0,1)	(1,1)

(0,0)	(0,0)
(1,0)	(0,0)

The Mutation stage is then applied as follows: for each chromosome (Matrix) every entry in the upper triangle is replaced with the XOR of itself with the symmetric of it in the lower triangle. So in our example let us take the 1st chromosome:

(0,0)	**(0,0)**
(0,1)	(1,1)

The entry in bold will be replaced with the result of its XOR with its symmetric in lower triangle of the matrix. In our case, (0,0) will be replaced by **(0,0)** XOR (0,1)= (0,1).

After mutating the above chromosomes we result in 4 new ones:

(0,0)	(0,1)
(0,1)	(1,1)

(0,0)	(0,0)
(1,0)	(0,0)

(0,0)	(1,1)
(0,1)	(1,1)

(0,0)	(1,0)
(1,0)	(0,0)

Now it is time to find the two survivals among the 4 chromosomes. The survivals are the most two fitter among these four chromosomes. To do so, a fitness function must be calculated for each one. The fitness function of each chromosome is the sum of the service cost and the transportation cost resolution from the corresponding combination.
For example, let us consider the 1st chromosome:

(0,0) (0,1)
(0,1) (1,1)

In this chromosome, the active sub-resources are: the sub-resource 2 of the resource (cost 0.1* $s_{12} = 4.5$), and the two sub-resources of the resource 2 (service cost 10.5, and 4 respectively). Then the total service cost resulting from this combination is 19.
The transportation cost is then calculated by considering the active sub-resources as independent supply points in a transportation problem. If the resulting supplying quantity does not satisfy the total demand, the chromosome is rejected. In the other case, the model is solved as a transportation problem, and its objective function is summed to the GA fitness function.

Thus, the two most fitted chromosome are considered as the next generation, and the procedure is repeated to create new chromosomes. In order to stop the repetition, GATRA provide two possibilities: either a threshold is achieved, of a convergence is reached (no change in chromosomes after some number of iterations.

The following figure presents a pseudo-code of the GATRA algorithm

Input:
- Number of supplies: N
- Number of destinations: M
- Number of resources: n
- Supply amount of each resource : S[n]
- Demand amount of each destination : D[M]
- Cost of service of each resource: C[n]
- Transportation cost from resources to each destination T[M][n]

Output:
- Optimal State Matrix for the resources (on/off).

Variables:
- *Originating_List*, the list the holds the parent chromosomes.
- *Convergence_index* = *100* , Genetic Algorithm's iterations stop whenever this convergence index is exceeded

---

1. **Initialize** randomly two chromosomes, each chromosome represents all states (on=1, off=0) for all sub-resource w.r.t each demand: *Originating_List(2)*
2. **Initialize** the Optimal Solution Matrix to be the output : *Optimal_Matrix*
3. **do Until** *Convergence_index* is greater than 100
4. *CrossOver(Originating_List)* to get 2 new born chromosomes, all 4 chromosomes are saved in *CrossOver_List*
5. **foreach** chromosome in the *CrossOver_List*
6. **do** Randomise a number *rand* between 0 and 100
7. **if** *rand* <50 then
8. **do** *Mutate(the_Current_Chromosome)* and add it to the *Mutated_List*
9. **else** keep the chromosome as it is without Mutation and add it to the *Mutated_List*
10. **foreach** chromosome *case* in the *Mutated_List*
11. **do if** *case* is Good
12. **do** Calculate the transportation matrix for *case*
13. **do** calculate the fitness for *case*
14. **do if** the result is less than the *Best_score*
15. **do** replace the new *Best_score*
16. **do** replace the new best chromosome with *case*
17. **do** reset *Convergence_index* to *0*
18. **do if** the *case* is better than one of the chromosomes in the *Originating_List*
19. **do** replace *case* in the *Originating_List*
20. **end do**
21. **return** *Optimal_State_Matrix*

4. Observations and Results

In this section, we state some results of experiments that we made using GATRA that measures the performance and accuracy of our algorithm.
All tests were made on a machine with an i3-2100 CPU @ 3.10GHz, a 12 GB DDR RAM, and a 64-bit Operating System – Service pack 1 (windows 7 Ultimate).

The first experiment consisted to measure the performance in term of execution time for the GATRA for certain scenarios. It was applied on 9 cases in which we started with 2 resources and 2 demands, to reach 10 resources and 10 demands, and we considered the number of resources to be constant (3) for all these scenarios.

In this experiment, we have considered the two stop criteria of the genetic algorithm: convergence line or a threshold (best score).
First, we have used the convergence line to choose when to stop execution, i.e. if the algorithm did not find better results in the next 300 iterations then we consider the latest (best score) recorded to be the final output.

The figure below shows the details of how this test went:

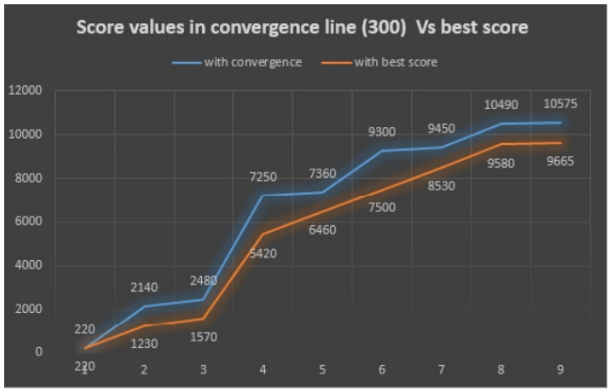
Resources	Sub-Resources	Demand	Number of iterations	Execution time in seconds	score
2	3	2	311	1.19	220
3	3	3	496	1.65	2140
4	3	4	225	1.17	2480
5	3	5	173	34.39	7250
6	3	6	373	2.14	7360
7	3	7	510	3.28	9300
8	3	8	203	2.64	9450
9	3	9	317	4	10490
10	3	10	361	5.32	10575

Second, we have achieved an experiment on the same input information, but we considered now to choose the best score method using a threshold, i.e. we declare the threshold we want to reach, then, GATRA will execute until it reaches this score (GOAL), and it was simple to choose the optimal score, since we considered our samples for testing to be intuitive cases to make tests more clear and fruitful.

So for each case we have chosen a best score and ran out test and the results came as follows:

Resources	Sub-resource	Demand	number of iterations	execution time in seconds	score
2	3	2	293	0.55	220
3	3	3	967	120.06	1230
4	3	4	91781	240.05	1570
5	3	5	6653	180.26	5420
6	3	6	29955	120.05	6460
7	3	7	798	3.58	7500
8	3	8	42708	232.8	8530
9	3	9	37544	255.76	9580
10	3	10	11729	95.54	9665

In comparing the two results we can see that as we move towards the "best score", GATRA consumes more iterations in some scenarios and thus consumes more execution time, though it can reach the best score in some scenarios when applying an "increased" convergence line technique. And the below figures shows the scores between the two techniques with varied convergence values:



As a result, as the demands increase, it's trivial that the cost increases, but within its minimal values (Optimal), and GATRA shows that as the cost increases, iterations decreases to reach 1 at the end when the Demand is Max and there is only one option, but it keeps almost a constant exec time average of 20.42 sec, to cover the convergence time (300).

These results have proved the efficiently of GATRA to solve such complex transportation models.

5. Conclusions

In this paper, we have presented a new technique (GATRA) to solve complex supplies/demands system, which combine the genetic algorithm with the transportation theory.

As a main conclusion, GATRA strongly proves that it can be a very powerful tool combing the power of GA and transportation problem and their functionalities. It provides a great analytic brain unit that can be embedded in any self-adaptive-distributed system, and it can be used in such systems to enhance their decision making through maximizing productivity and minimizing the cost.

Being in its first versions, GATRA provides promising results through its low execution time and optimal results which proves to be conforming the minimal results in the intuitive cases examples. GATRA can be enhanced and refined through advancing its main areas like the crossover and the mutation functions, which may decrease its execution time, making it more efficient perhaps in more complex and huge systems.

As shown in this paper, GATRA is designed to deal with a specified cost case, so it is possible to be expanded to cover all cost cases. This can be done by adding new modules to it which would be considered simpler than the current main module. GATRA can be also integrated with big data supply systems which have the same nature, to empower them. We can also add to it a more detailed input module to allow serving more constraint requests such as, let's say an "off schedule" for sum resources, a fixed resources combination, or when some transportation roads are not used (maintenance or similar cases).

References

[1] Ababei, Cristinel, and Rajesh Kavasseri. "Efficient network reconfiguration using minimum cost maximum flow-based branch exchanges and random walks-based loss estimations." IEEE Transactions on Power Systems 26.1 (2011): 30-37.
[2] Banzhaf, Wolfgang. Foundations of genetic algorithms. Morgan Kaufmann Publishers Inc., 1999.
[3] Chiang, H-D., and Rene Jean-Jumeau. "Optimal network reconfigurations in distribution systems. II. Solution algorithms and numerical results." IEEE transactions on Power delivery 5.3 (1990): 1568-1574.
[4] Chouhan, Sridhar, et al. "Intelligent reconfiguration of smart distribution network using multi-agent technology." Power & Energy Society General Meeting, 2009. PES'09. IEEE. IEEE, 2009.
[5] T. Dean and L. Greenwald. A formal description of the transportation problem. Technical Report CS-92-14, CS Dept Brown Univ., March 1992.
[6] Frutos, Mariano, et al. "Comparison of multiobjective evolutionary algorithms for operations scheduling under machine availability constraints." The Scientific World Journal 2013 (2013).
[7] Hsu, Fu-Yuan, and Men-Shen Tsai. "A non-dominated sorting evolutionary programming algorithm for multi-objectives power distribution system feeder

reconfiguration problems." International Transactions on Electrical Energy Systems 23.2 (2013): 191-213.

[8] Kalambe, Shilpa, and Ganga Agnihotri. "Loss minimization techniques used in distribution network: bibliographical survey." Renewable and Sustainable Energy Reviews 29 (2014): 184-200.

[9] Mahmoodabadi, M. J., M. Taherkhorsandi, and A. Bagheri. "Pareto design of state feedback tracking control of a biped robot via multiobjective PSO in comparison with sigma method and genetic algorithms: modified NSGAII and MATLAB's toolbox." The Scientific World Journal 2014 (2014).

[10] Messac, Achille. "Physical programming-effective optimization for computational design." AIAA journal 34.1 (1996): 149-158.

[11] Morton, Anthony B., and Iven MY Mareels. "An efficient brute-force solution to the network reconfiguration problem." IEEE Transactions on Power Delivery 15.3 (2000): 996-1000.

[12] Pop, Petrica, et al. "A hybrid based genetic algorithm for solving a capacitated fixed-charge transportation problem." Carpathian Journal of Mathematics (2016): 225-232.

[13] Rao, R. Srinivasa, et al. "Power loss minimization in distribution system using network reconfiguration in the presence of distributed generation." IEEE transactions on power systems 28.1 (2013): 317-325.

[14] Ren, Hongbo, et al. "Multi-objective optimization for the operation of distributed energy systems considering economic and environmental aspects." Applied Energy 87.12 (2010): 3642-3651.

[15] Santos, A. C., et al. "Node-depth encoding and multiobjective evolutionary algorithm applied to large-scale distribution system reconfiguration." IEEE Transactions on Power Systems 25.3 (2010): 1254-1265.

[16] Sudhakar, T. D., and K. N. Srinivas. "Restoration of power network–a bibliographic survey." International Transactions on Electrical Energy Systems 21.1 (2011): 635-655.

[17] Tomoiagă, Bogdan, et al. "Pareto optimal reconfiguration of power distribution systems using a genetic algorithm based on NSGA-II." Energies 6.3 (2013): 1439-1455.

[18] Vogel, Curtis R. Computational methods for inverse problems. Society for Industrial and Applied Mathematics, 2002.

.