SREE VIDYANIKETHAN ENGINEERING COLLEGE

(Affiliated to Jawaharlal Nehru Technological University Anantapur)
Sree Sainath Nagar, A. Rangampet, Tirupati – 517 102, Chittoor Dist., A.P.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project Work entitled

"Using Optimization to improve picking routes"

is the bonafide work done by

K C KISHORE KUMAR	**15121A0584**
K S D N V S CHANDU RAM	**15121A0579**
K R SAI KUMAR	**15121A05A3**
K SIVAJI NAIK	**15121A05A0**
K BHAVANA	**15121A05B7**

In the Department of Computer Science and Engineering, Sree Vidyanikethan Engineering College, A. Rangampet. is affiliated to JNTUA, Anantapuramu in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering during 2015-2019.

This is work has been carried out under my guidance and supervision.

The results embodied in this Project report have not been submitted in any University or Organization for the award of any degree or diploma.

Internal Guide

Mr. Shaik Salam M.E.,(Ph.D)
Associate Professor
Dept of CSE
S.V.E.C
A.RANGAMPET

Head

Dr. M. Sunil Kumar
Prof & Head
Dept of CSE
S.V.E.C
A.RANGAMPET

INTERNAL EXAMINER

EXTERNAL EXAMINER

"USING OPTIMIZATION TO IMPROVE PICKING ROUTES"

A Project Report submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNVERSITY ANANTAPUR.

In Partial Fulfillment of the Requirements for the Award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING
BY

K C KISHORE KUMAR	**15121A0584**
K S D N V S CHANDU RAM	**15121A0579**
K R SAI KUMAR	**15121A05A3**
K SIVAJI NAIK	**15121A05A0**
K BHAVANA	**15121A05B7**

Under the Guidance of

Mr. Shaik Salam M.E.,(Ph.D)
Associate Professor
Dept of CSE, SVEC

Department of Computer Science and Engineering
SREE VIDYANIKETHAN ENGINEERING COLLEGE
(Affiliated to JNTUA, Anantapuramu)
Sree Sainath Nagar, Tirupathi – 517 102

2015-2019

DECLARATION

We hereby declare that this project report titled **"USING OPTIMIZATION TO IMPROVE PICKING ROUTES"** is a genuine project work carried out by us, in **B.Tech** ***(Computer Science and Engineering)*** degree course of **Jawaharlal Nehru Technological University Anantapur** and has not been submitted to any other course or University for the award of any degree by us.

Signature of the student

K C KISHORE KUMAR	**15121A0584**
K S D N V S CHANDU RAM	**15121A0579**
K R SAI KUMAR	**15121A05A3**
K SIVAJI NAIK	**15121A05A0**
K BHAVANA	**15121A05B7**

ACKNOWLEDGEMENT

We are extremely thankful to our beloved Chairman and founder **Dr. M. Mohan Babu** who took keen interest to provide us the infrastructural facilities for carrying out the project work.

We are highly indebted to **Dr. P.C. Krishnamachary**, Principal of Sree Vidyanikethan Engineering College for his valuable support and guidance in all academic matters.

We are very much obliged to **Dr. M. Sunil Kumar,** Professor & Head, Department of CSE, for providing us the guidance and encouragement in completion of this project.

We would like to express our indebtedness to the project coordinator, **Mr. Shaik Salam**, Associate Professor, Department of CSE for his valuable guidance during the course of project work.

We would like to express our deep sense of gratitude to the guide **Mr. Shaik Salam**, Associate Professor, Department of CSE, for constant support and invaluable guidance provided for the successful completion of the project.

We are also thankful to all the faculty members of CSE Department, who have cooperated in carrying out our project. We would like to thank our parents and friends who have extended their help and encouragement either directly or indirectly in completion of our project work.

ABSTRACT

Slotting is the activity of determining the most appropriate storage location for each item in your warehouse. In slotting we make sure those things we use the most frequently are the most easily accessible. In doing so, the things that we don’t use frequently will often naturally work their way back into those less accessible areas (slots). The items in your warehouse each have their own unique combination of characteristics. The most important characteristics is the frequency of physical touches for each specific item. As physical touch activities go, the one that gets the most attention is order picking. Successful warehouse slotting maximizes the use of available space within a warehouse through more efficient storage and picking, while reducing handling costs. Picking is a logistic warehouse's processes which involves taking and collecting articles in a specified quantity before shipment to satisfy customers' orders. It is a basic warehousing process and has an important influence on supply chain's productivity. Warehouse optimization is key to the efficient operation of warehouses of all sizes. A disciplined process, warehouse optimization includes automation and a determination of how to save time, space, and resources while reducing errors and improving flexibility, communication, management, and customer satisfaction. The problem of storage space optimization system based on different demand frequencies and different demand quantities.

INDEX

LIST OF FIGURES

LIST OF TABLES

1.INTRODUCTION

1.1 OVERVIEW

Generally in a warehouse there are many items that are stored. These items may be finished products or even raw materials. So whenever there is any order for items these items need to be picked from the warehouse and delivered. In order to do this the warehouse needs to be travelled to pick the required items. As the warehouse is big and we have huge possibilities that the required items will be in different locations in the warehouse we have problems regarding which item to pick after picking one item. So in order to solve this problem we determine an optimal path through which we can pick all the required items in a single journey. This process is called routing. Using this process a path which uses less time to pick all the items can be determined. So for doing this routing, we use genetic algorithm. A Genetic Algorithm is a population based search and Optimization method that mimics the process of natural evolution. This algorithm takes the distance between the items in the warehouse and calculates the fitness scores there by prioritizing the path. Here a population size will be taken and based on it no. of chromosomes are produced. The fitness scores are compared between and then these chromosomes are mated to produce the child chromosomes. Then the child chromosomes are further mated in a process called mutation and the process continues

and then we get a final chromosome which will be our desires optimal path.

1.2 STATEMENT OF THE PROBLEM

In the future it is expected that robots can do the same activities as the operators in a traditional warehouse. So, one could have a warehouse where both robots and operators are responsible for replenishment and picking tasks. The assignment is to make a system where we can learn from the operators in the WH (how is the slotting and picking routes operator uses). The idea is to use artificial intelligence, that learns from operators, to develop a strategy which will be used to instruct Robots.

1.3 OBJECTIVES

There are certain objectives that are satisfied on picking these routes. They are:

1. **Minimize Time:** This is the main objective of the routing process. The time taken for picking items should be lessened by determining effective and optimal path to pick the items.
2. **Minimize work load:** Determining an effective route for picking items reduces the workload on the workers and even the transport units. By predefined paths the workers need not think about in which path to go in the warehouse thereby decreasing the mental thinking.
3. **Increase in picking efficiency:** There should be a significant increase in picking efficiency. The no. of items picked with in a certain amount of time should increase.
4. **Minimize distance travelled:** The distance travelled with in the warehouse should be minimized. The picker should be able to pick the shortest path in order to pick the item that are required.

1.4 SCOPE

Proposed System:

1.This project is to construct an efficient route picking system to determine the shortest possible path in a warehouse.

2. In this the admin can simply give items that needs to be delivered from the warehouse.

3.After that our system prepares the picking list and it wil be used to pick the required items by the picker.

4.It works based on the genetic algorithm.

1.5 APPLICATIONS

There are several applications for the genetic algorithm we are using here.

They are:

Robotics

Robotics involves human designers and engineers trying out all sorts of things in order to create useful machines that can do work for humans. Each robot's design is dependent on the job or jobs it is intended to do, so there are many different designs out there. GAs can be programmed to search for a range of optimal designs and components for each specific use, or to return results for entirely new types of robots that can perform multiple tasks and have more general application. GA-designed robotics just might get us those nifty multi-purpose, learning robots we've been expecting any year now since we watched the Jetsons as kids, who will cook our meals, do our laundry and even clean the bathroom for us!

Automotive Design

Using Genetic Algorithms [GAs] to both design composite materials and aerodynamic shapes for race cars and regular means of transportation (including aviation) can return combinations of best

materials and best engineering to provide faster, lighter, more fuel efficient and safer vehicles for all the things we use vehicles for. Rather than spending years in laboratories working with polymers, wind tunnels and balsa wood shapes, the processes can be done much quicker and more efficiently by computer modeling using GA.

Computer Gaming

Those who spend some of their time playing computer Sims games (creating their own civilizations and evolving them) will often find themselves playing against sophisticated artificial intelligence GAs instead of against other human players online. These GAs have been programmed to incorporate the most successful strategies from previous games – the programs 'learn' – and usually incorporate data derived from game theory in their design.

DNA Analysis

GAs have been used to determine the structure of DNA using spectrometric data about the sample.

1.6 LIMITATIONS

- Repeated fitness function evaluation for complex problems is often the most limiting segment of genetic algorithms. Finding the optimal solution to complex high-dimensional problems often requires very expensive fitness function evaluations. In real world problems, a single function evaluation may require several hours to several days of complete simulation.
- Genetic algorithms do not scale well with complexity. That is, where the number of elements which are exposed to mutation is large there is often an exponential increase in search space size. This makes it extremely difficult to use the technique on problems such as designing an engine, a house or plane.
- The "better" solution is only in comparison to other solutions. As a result, the stop criterion is not clear in every problem.
- For problems with more amounts of input data, it becomes difficult in order to determine or set a stop criterion because it takes more time for the processes like mutation, selection and crossover. With the complexity in data the complexity in the solution increases.

2.LITERATURE SURVEY

There are numerous studies in the field of order picking and warehousing. To present classifications of order picking systems and routing heuristics for single block warehouses, review of [1], which is about manual order picking processes regarding layout design, storage assignment techniques, order batching, zoning, routing strategies, order accumulation and sorting methods, is analyzed. A taxonomy developed by [2] is also considered to classify order picking systems into five categories as "Picker-to-parts", "Pick-to-box", "Pick-and-sort", "Parts-to-picker" and "Completely automated picking". [3] provides a literature review about order picking systems as well. Categorization of order picking systems, components of order picking time, factors affecting order picking process and routing heuristics are topics covered by the survey which is broadly used for the introduction part of this study.

This study is focused on improving order picking performance of a multi-block warehouse so a performance basis for multi-block warehouses is needed to compare the proposed solution. To analyze solution approaches for multi-block warehouses, various studies for multi-block warehouses are examined. [4] considers a parallel-aisle warehouse. Average travel time is compared for warehouses with and without a middle aisle, through a simulation. Three factors are taken into account: warehouse size, warehouse layout (the presence or absence of a middle aisle; the number of aisles), pick list size. A routing algorithm is proposed where aisle changing is allowed, due to cross aisles. Also [5] introduces combined and combined+ routing heuristics in this study for warehouses with more than two cross aisles where items are stored

randomly. The proposed heuristics use dynamic programming. The performance of the proposed heuristics is compared to a branch-and-bound algorithm under different warehouse layouts and pick list sizes. [6] studies order picking problem and compare optimal and heuristic algorithms, in terms of average travel time and total route time (which includes other activities such as administrative tasks, acquisition and dropping off pick carriers, in addition to travel time). [7] describes routing policies and storage assignment policies for multi-block warehouses. A simulation study takes place to evaluate several storage assignment policies and routing policies for various layouts.

To analyze different management decision problems that warehouse managers encounter to improve warehouse productivity, reviews and studies based on main warehouse activities and policies are examined. [8] mention basic warehouse functions, order picking strategies, automation, classification of order picking systems and warehouse equipment. [9] presents a literature review including classification of warehouses, strategic-tactical-operational warehouse decisions, storage location assignment problem, order batching, routing and sequencing issues. Storage location assignment is considered as an intermediate range management decision whereas routing, sequencing and order batching are considered as short range operational decisions. [10] reviews literature on warehouse design and control systems, focusing internal warehouse structure and operations. Warehouse characteristics regarding warehouse processes, warehouse resources, warehouse organization issues are covered. Warehouse design problems at strategic, tactical and operational level are examined. [11] presents a

review and categorize operational level warehouse operation planning problems, based on four warehouse activities: receiving, storage, order picking and shipping. [12] compares several picking, storage and routing policies in manual order picking systems via a simulation study. The effect of these three decisions are examined on order picker travel time, with regards to reduction in total picking time by comparing to a baseline policy. The baseline policy refers to the actual policy of the firm which employs traversal routing and random storage. A sensitivity analysis is also conducted to explore the effect of order size, warehouse shape, location of pick-up/drop-off point, and demand distribution on order picker travel time.

[13] focusses joint order batching and order picking problems. A mathematical formulation is proposed for the joint problem, based on integrating bin packing problem and TSP. It is stated that bin packing problem is equivalent to order batching problem where bin represents the order picking vehicle and items to be packed are items to be picked. The aim is to find the assignment of orders to batches to minimize the number of batches subject to not exceeding the capacity of the vehicle. TSP is equivalent to obtaining the sequence of items to be picked. Moreover, two heuristic algorithms are suggested to be able to solve the problem within reasonable running time. This is because order batching and order picking problems are said to be two key operational problems which must be solved frequently and require fast solutions. [14] considers order picking problem where items can be stored in multiple locations, as opposed to general setting. A model is suggested for simultaneous determination of location assignments and picking

sequence. However, given the complexity of the problem, TSP heuristics such as nearest neighbor and shortest arc are modified for the problem setting. Also, a tabu search algorithm is developed for the problem.

In terms of studies that describe and implement GA, [15] examines evolutionary algorithms, in specific, genetic algorithms. Steps and process of genetic algorithm, genetic operators, advantages of genetic algorithms and selection methods are explained. This study is mainly used to have a brief introduction to GA and to understand the functioning of GA. Among diverse applications of GA to solve different problems in literature, studies proposed for warehousing problems using GA are concerned majorly. To be able to decide on selection method, crossover and mutation operators and to determine the value ranges of parameters to be used in GA in parameter tuning for crossover and mutation probability, the following studies are examined. [16] considers order picking problem in an automated single-block warehouse by taking travelling time into account. A GA is implemented which uses roulette wheel as selecting strategy, with the optimal individual preserving strategy (in other words, elitism). Solutions' convergence situation in terms of total travel time is reported under different iteration times. [17] considers order batching problem where customer orders are grouped into batches optimally to minimize total travel distance by the help of a GA. The proposed algorithm can be applied to not only single-aisle or rectangular but any type of warehouse layout. [18] proposes an order batch picking model which takes earliness and tardiness penalties into account, in addition to travel cost. Retrieving items earlier than their due date leads to piled up items around the

warehouse. Retrieving items after their due date leads to transportation delays and customer dissatisfaction. Roulette wheel selection approach is employed. A multiple-GA method, which consists of two different GA-based algorithms, is constructed to solve the proposed model. The first GA algorithm is used to find an optimal order batching plan to minimize earliness and tardiness penalties and travel cost. The second GA algorithm is utilized to obtain the optimal travel path within an order batch to minimize travel distance. A parameter tuning takes place to find best parameter combinations,regarding maximum number of iterations, crossover probability, mutation probability and population size. [19] focuses on order batching problem in a low-level, picker-to-parts, single-block rectangular warehouse. A GA is presented which applies parameterized uniform crossover operation that mixes the information of two parents according to a fixed mixing ratio. An immigration operator is employed, instead of a mutation operator, by generating some new chromosomes as initial solution generation phase in order to provide genetic diversity and not to get stuck in local optima. [20] considers storage allocation and order picking problems in a single-block fast-moving consumer goods warehouse. Elitism is implemented by copying the first best two chromosomes to the next generation. [21] focuses on order picking multi-objective optimization problem. The number of order pickers per shift and the best retrieving sequence of items are to be determined. Roulette wheel is used as a parent selection technique and order crossover is employed to generate offspring. Inversion mutation is incorporated. [22] proposes a dynamic mathematical model to solve small-size order picking problems by

taking product life, customer importance, probabilistic demand and backordering strategy into account. A GA is proposed to solve similar large-size problems. Three metrics are used for performance comparison of two methods: Elapsed CPU time, number of fulfilled orders, quality of objective function. [23] examines order picking problem in a multi-aisle automated warehouse to minimize total travel time of storage/retrieval machine. In the warehouse, each item can be retrieved from several storage locations. A GA is constructed which uses roulette wheel as selection method and partially matched crossover as crossover operator. Two performance measures, CPU time and travel time, are analyzed. [24] considers the problem of a film-copy deliverer, as an extension of TSP. There is an analogy between order picking problems and TSP such that order picker is analogous to salesman whereas items to be picked are analogous to cities to be visited. Therefore, film-copy delivery problem is also similar to order picking problem. In this problem, there are several cinemas and only one film-copy. The duty of the deliverer is to bring the film-copy to each cinema based on the predetermined show times. A GA is developed for which a new crossover operator is designed to prevent illegal offspring and a new mutation operator is designed to mutate offspring. [25] investigates the relationship between order picking problem inside warehouse and vehicle routing problem outside warehouse for conventional single-block and multiple-cross aisle warehouses. A GA-based approach is proposed to solve these problems in a hierarchical manner. [26] solves order batching and pick routing problems simultaneously via GA based methods. Two new GA-based methods are

proposed that can be employed for both conventional single block and multi-block warehouses. Roulette wheel is used as selection method.

3.ANALYSIS

Reliability:

Reliability is the degree to which an assessment tool produces stable and consistent results. Types of Reliability. Test-retest reliability is a measure of reliability obtained by administering the same test twice over a period of time to a group of individuals.

Flexibility:

Flexibility is the ease with which the system can be reused, deployed, and tested.

Performance:

The system must be interactive and the delays involved must be less. So in every action-response of the system, there are no immediate delays. In case of opening windows forms, of popping error messages and saving the settings or sessions there is delay much below 2 seconds, in case of obtaining outputs there should be quick evaluation and other operations to be performed.

Security:

There will be no major security breaches for the system as it will be used mainly by the professionals and not everyone in the system. They should be careful in order to prevent the data from leaking outside. They should be also careful regarding any hacks that may occur.

Data Analysis:

The main data here in our picking system is the name of our items and their locations in the warehouse. The locations will be in form of co-ordinate points.

ITEMS	X COORDINATE	Y COORDINATE	PRICE(each in Rs.)	QUANTITY
Watches	50	35	1000	300
Televisions	37	39	40000	100
Bags	26	35	500	500
Books	4	6	500	1000
Mobiles	2	8	15000	300
Laptops	7	5	45000	200
Refrigerators	9	12	50000	100
Air Conditioners	42	36	35000	100
PenDrives	3	5	2000	400
HDD	24	37	6000	250
Flash Cards	45	20	5000	150
Bottles	35	8	300	675
Fans	9	24	1000	570
Ear Phones	5	24	1500	500
Electic Bulbs	8	8	150	450
Deodrants	33	26	700	400
Vaccum Cleaners	36	40	5000	150
Sandals	43	25	300	850
Cameras	29	24	25000	225
Power Banks	46	27	5000	200
Blankets	25	37	300	700
Hair Dryers	18	26	2000	500
Trimmers	22	24	1000	600
Soaps	18	20	50	5000
Washing Machines	29	35	40000	150
Shoes	20	41	2000	1000
Goggles	30	42	1000	1050
Kerchiefs	35	45	5	900
Cosmetcis	44	21	600	700
Shirts	15	17	1000	4000
Pants	12	23	800	4000
Chairs	23	32	100	600
Tables	24	39	500	250

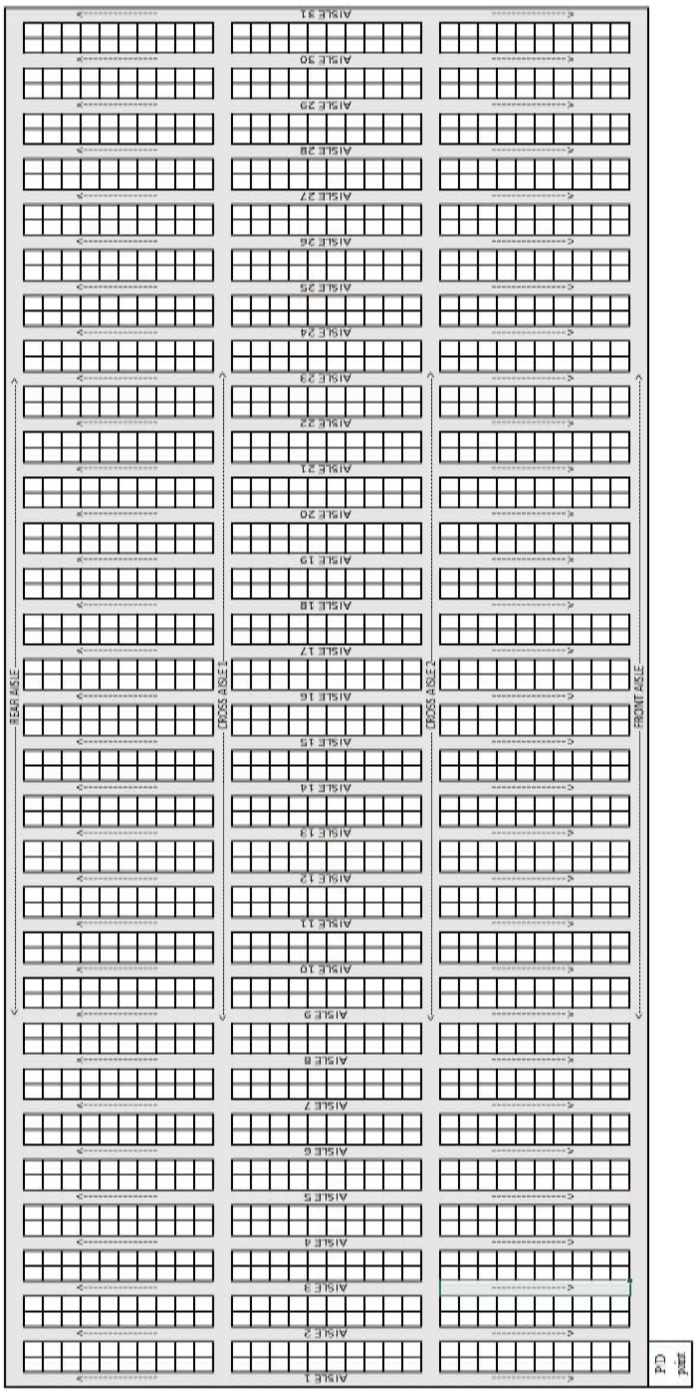
3.1 Available Data in Ware House

Here in our system we assume the warehouse as a co ordinate system and represent the points with the items with points. In the above mentioned datasets the x coordinates and y coordinates represent the ones on the X-axis and Y-axis respectively. The prices and the quantity of the items are also mentioned.

4.DESIGN

The main functionality of a warehouse is to receive and distribute products all along the supply chain process. So, it becomes important to take into consideration how the warehouse is designed. The requirements of the present as well as for the future should always be kept in mind. The goal of warehouse designing is to optimize the warehousing operations and achieve maximum efficiency. In short, the warehouse design element aims to maximize the utility of space, equipment, and efficiency of operations.

So in a warehouse we have different racks arranged in row and column wise and the items are arranged in particular order as per the usage and the space. The space in between the racks where we travel and can pick the items is called an aisle. So the space adjacent to the P/D point is called the front aisle and the back aisle is called as rear aisle. The aisle between the two column of racks will be called as cross aisle. The pick or drop point is the inlet or the exit from the warehouse. The P/D point is where we start to go near the racks of the items and pick the desired items. From here through the aisles we look for the desired items and fetch them.

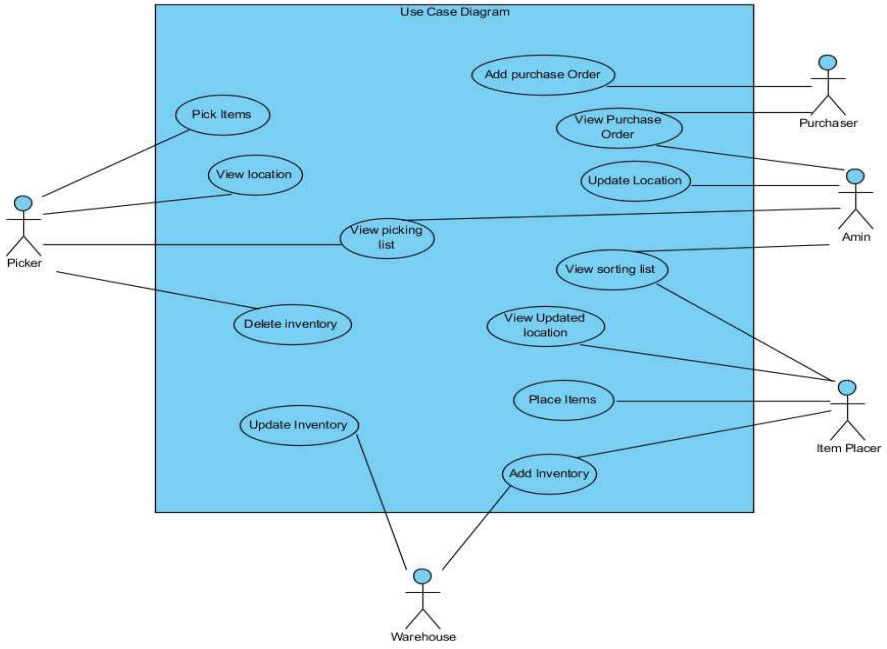


4.1 Warehouse Layout

Use case Diagram:

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. It represents different actors that take part in the system and their respective activities. The different actors in the order picking system are:

1. Purchaser 2. Admin 3. Picker 4. Item Placer
1. Warehouse

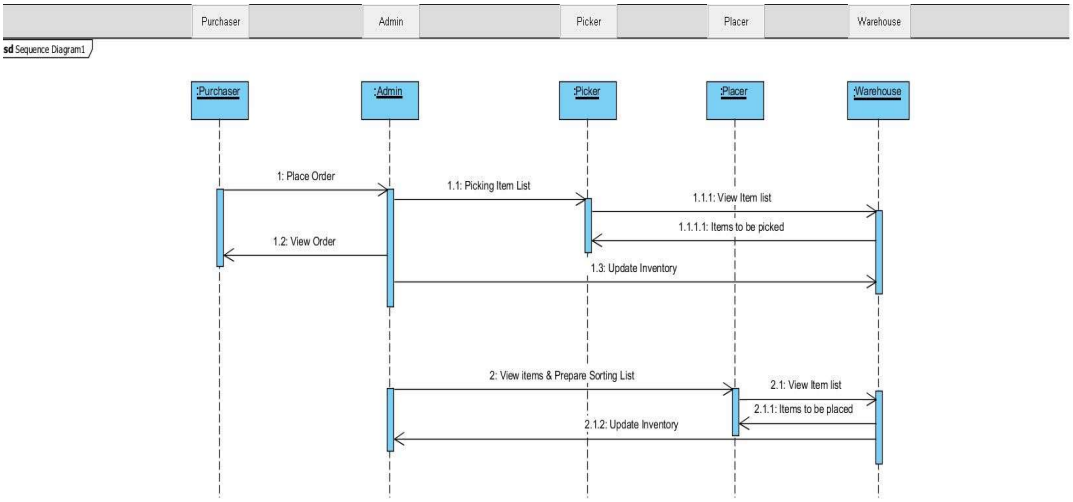


4.2 Use Case Diagram

1. **Purchaser:** Purchaser is the one who buys things from a buyer. He adds the purchase order based on his desire and this will be viewed by the admin. The purchaser can also view the order he placed.
2. **Admin:** Admin plays a key role in the warehouse picking and placing operations. He is the one who looks after the purchase orders and prepares respective picking or placing lists based on the orders. If there is a need for delivery then the admin looks after all the orders that have been placed and prepares the picking list and he even prepares a slotting list if new items enter the warehouse. He updates the places of the items after each delivery.
3. **Picker:** He is the one who is responsible for the picking activities in the warehouse. He receives a picking list from the admin based on the delivery and picks the required items mentioned items in the list. In this way he does the picking activity.
4. **Item Placer:** He is responsible for the placing of the items in the warehouse. After every picking or if new items enter into the warehouse the admin updates the locations of the items and based on the new item locations he places the items.
5. **Warehouse:** This forms the warehouse of the system. This involves use cases like AddInventory, DeleteInventory. Whenever there are any new items to be placed into the the warehouse then the AddInventory is used and whenever there is any delivery then items are removed, then DeleteInventory is used.

Sequence Diagram:

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. The actors from the use case diagram form the objects in the sequence diagram.



4.3 Sequence Diagram

The Sequence diagram for picking items in a warehouse involves:

- **Place Order**: Here the purchaser will place an order based on his desire and this what is called placing an order.

- **Picking Item list:** Based on the order the picking list is prepared and given to the picker.
- **View Item list:** The list will be viewed by the picker to know which items need to be picked.
- **Items to be picked:** Then based on the list required items will be picked by the picker.
- **Update Inventory:** After picking the items the inventory will be updated that is the items that are left and the new item locations will be updated.
- **Prepare sorting list:** If new items are to be placed the admin creates a placing list which will be used by the item placer.
- **View Item list:** The list will be viewed to know which items are to be placed in which area and the items are placed.

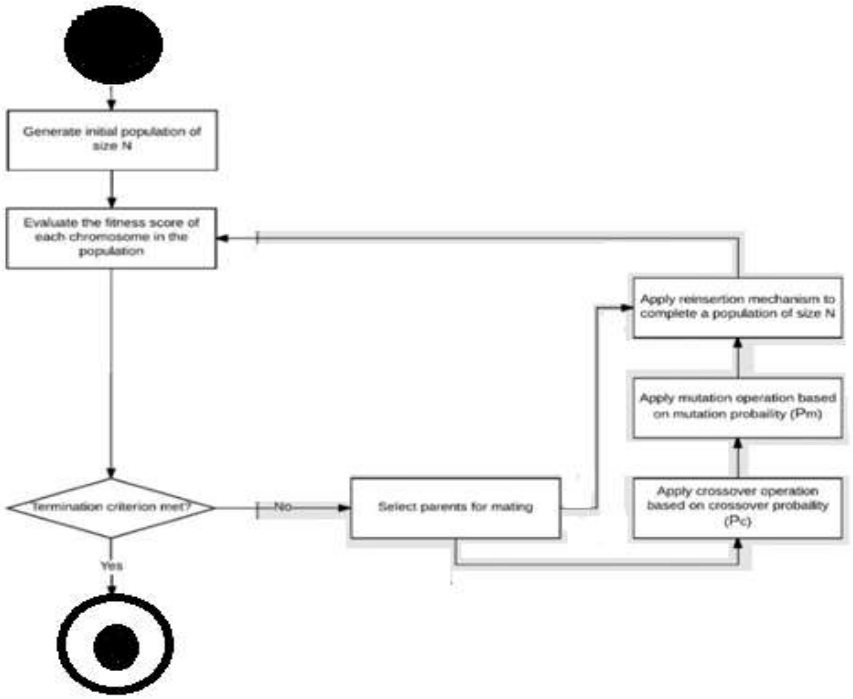
At last after all the operations of a particular delivery the inventory is again updated.

In this way in a sequence the operations in a warehouse are performed and the purchaser can view his order till he receives the items he ordered.

Activity Diagram:

This diagram explains the steps that are in the genetic algorithm. They are:

1. Selection
2. Crossover
3. Mutation
4. Reinsertion



4.4 Activity Diagram

5.IMPLEMENTATION

GA($Fitness$, $Fitness\_threshold$, p, r, m)

Fitness: A function that assigns an evaluation score, given a hypothesis.

$Fitness\_threshold$: A threshold specifying the termination criterion.

p: The number of hypotheses to be included in the population.

r: The fraction of the population to be replaced by Crossover at each step.

m: The mutation rate.

- *Initialize population:* $P \leftarrow$ Generate p hypotheses at random
- *Evaluate:* For each h in P, compute $Fitness(h)$
- While $[\max_h Fitness(h)] < Fitness\_threshold$ do

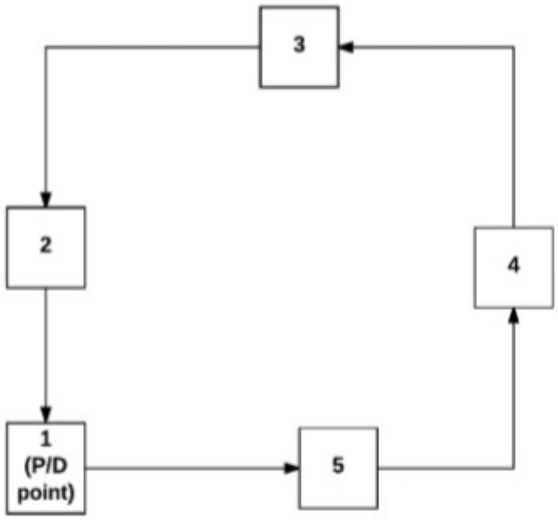
  Create a new generation, P_S:

  1. *Select:* Probabilistically select $(1-r)p$ members of P to add to P_S. The probability $\Pr(h_i)$ of selecting hypothesis h_i from P is given by

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

  2. *Crossover:* Probabilistically select $\frac{r \cdot p}{2}$ pairs of hypotheses from P, according to $\Pr(h_i)$ given above. For each pair, $\langle h_1, h_2 \rangle$, produce two offspring by applying the Crossover operator. Add all offspring to P_s.
  3. *Mutate:* Choose m percent of the members of P_s with uniform probability. For each, invert one randomly selected bit in its representation.
  4. *Update:* $P \leftarrow P_s$.
  5. *Evaluate:* for each h in P, compute $Fitness(h)$
- Return the hypothesis from P that has the highest fitness.

5.1 Genetic Algorithm



5.2 Path

	1 (P/D point)	2	3	4	5
1(P/D point)	0	10	12	8	15
2	10	0	7	11	9
3	12	7	0	18	13
4	8	11	18	0	6
5	15	9	13	6	0

5.3 Sample shortest distance matrix

Selection: In this step, we determine the chromosomes that are required to be mated to produce the offsprings. These are determined with the help of fitnesses of each chromosome. The fitness is determined with the help of distances between each item in the warehouse.

Chromosome	Fitness score	Probability of being selected	Contiguous Intervals
15432	0.01785714	0.24029	[0 – 0.24029]
32451	0.01960784	0.26385	[0.24030 – 0.50414]
24513	0.01960784	0.26385	[0.50415 – 0.76799]
43521	0.01724138	0.23201	[0.76800 – 1]
Total	**0.07431421**	**1**	

5.4 Selection Probabilities

Fitness = 1/(Distance travelled)

If the fitness is less than the threshold we select them for further process.

Random number	Fallen mapped interval	Corresponding chromosome to be selected (Mating pool)
0.638425	[0.50415 – 0.76799]	24513
0.48754	[0.24030 – 0.50414]	32451
0.045572	[0 -0.24029]	15432
0.439298	[0.24030 – 0.50414]	32451

5.5 Mating Pool Generation

1. **Cross Over**: Crossover is combining pairs of parents to create new individuals or offspring. Two children are produced from two parents through a crossover operator which is a function that gets some of genes from one parent and the rest from the other parent to form two offsprings.

Chromosome pairs in mating pool	Crossover Probability	Random number	Selected for Mating?
(24513 & 32451)	0.7	0.214939833	0.214939833 <= 0.7 ➔ **Yes**
(15432 & 32451)	0.7	0.790672194	0.790672194 > 0.7 ➔ **No**

5.6 Crossover Probability on the Mating Pool

3.Mutation:

Mutation is introducing new genetic material in the population by randomly altering some selected genes of offspring chromosomes generated by crossover operation.

Mated Chromosomes	Mutation Probability	Random number	Selected for Mutating?
34521	0.01	0.394763129	0.394763129 > 0.01 ➔ **No**
52413	0.01	0.008672194	0.008672194 <= 0.01 ➔ **Yes**

5.7 Implementation of Mutation

From one generation to another, population size should remain fixed.

4.Reinsertion :

The difference between population size and the number of chromosomes reproduced through evolution is termed a generation gap.

Reinsertion mechanism is used to decide which chromosomes should survive and be inserted to the next generation, in case of a generation gap.

Chromosomes in mating pool	Fitness score	Rank of chromosome	Selected for Reinsertion?
24513	0.01960784	1	**Yes**
32451	0.01960784	1	**Yes**
15432	0.01785714	3	**No**
32451	001960784	1	**Yes (duplicated)**

5.8 Implementation of Re-Insertion

Having a new generation ready, GA evaluates each chromosome of the new generation using fitness function.

6.EXECUTION PROCEDURE AND TESTING

In GeneticRoutines.py file we implemented the selection, mutation and crossover methods which in turn returns the offspring which is used as chromosome in next generation.

In TSP.py file we have taken the distances between items and the corresponding locations of items in string and tuple format using Itemloc class and to find the route using Route class.

In mod-Graphic.py we visualized the how picking of items done in a ware house based on the locations using pygame and metrics like performance of algorithm on local maximum values. Using tkinter we implemented Graphical user interface (GUI) to pick the available items in a ware house by selecting the required available items.

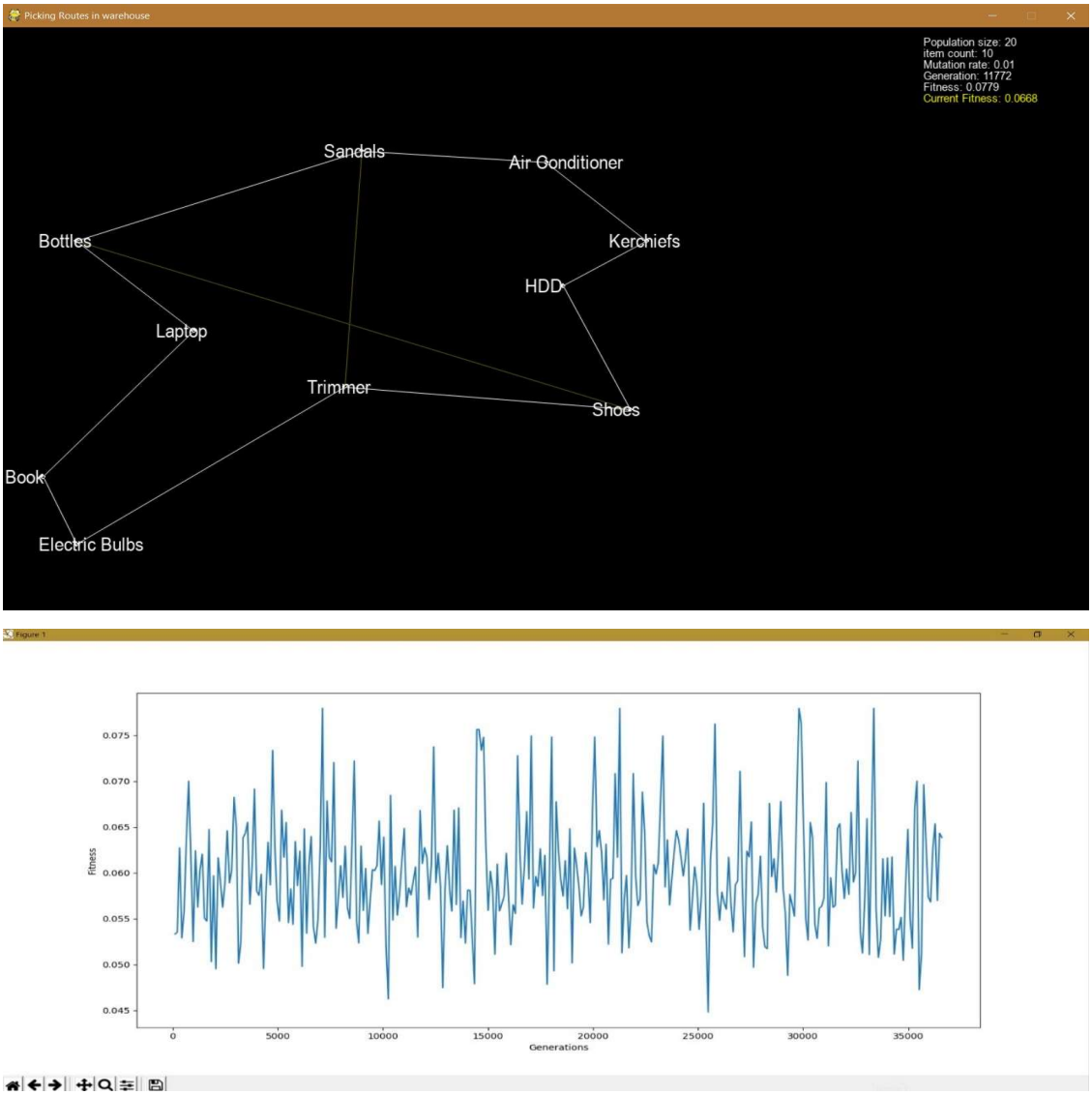
On selecting the required items Genetic algorithm runs based on the number of generations or it can be limited on time basis. Suppose picker selects 10 items on various locations in a ware house then algorithm running can be limited by generations for example we have to limit the generations to 10,000 or for example we have to limit the algorithm for 60 seconds.

As picking items increases the number of generations generated per second decreases. For that purpose it is good to limit algorithm on number of generation basis.

7. RESULTS & PERFORMANCE EVALUATION

7.1 User Interface

7.2 Selecting Items



7.3 Graphical Result

```
 RESTART: C:\Users\kumar\Desktop\project\modified Project\Genetic-Algorithm-TSP-Picking\mod-Graphic.py
pygame 1.9.4
Hello from the pygame community. https://www.pygame.org/contribute.html
>>> Selected Items: Book, Laptop, Air Conditioner, HDD, Bottles, Electric Bulbs, Sandals, Trimmer, Shoes, Kerchiefs
Best route: Trimmer->Electric Bulbs->Book->Laptop->Bottles->Sandals->Air Conditioner->Kerchiefs->HDD->Shoes
Best fitness: 0.07793949931605255
```

7.4 Output Path

8. CONCLUSION & FUTURE WORK

In the existing system in the warehouse picking there are robots that can pick the items in the warehouse but there is no proper way to find out an optimal and minimal path to follow in picking those items.

Our warehouse picking system provides an optimal path by using the genetic algorithm and thereby providing an optimal path. It uses the distances between the items to provide a path. But we know if there are more number of items then it will be difficult for the fitness functions to process and it will take more time to provide a result. So the future work to be done is to work.

APPENDIX

CODE

GeneticRoutines.py - C:\Users\kumar\Desktop\project\Genetic-Algorithm-TSP-Picking\libs\GeneticRoutines.py (3.6.2)

File Edit Format Run Options Window Help

```
from random import random, randint
from TSP import Route as Chromosome

def selection(pop):
    i = -1
    rand = random()
    while rand > 0:
        i += 1
        rand -= pop[i].chance
    return pop[i]


def mutation(population, rate):
    for chromosome in population:
        if random() < rate:
                r1 = randint(0, len(chromosome.genes) - 1)
                r2 = randint(0, len(chromosome.genes) - 1)
                chromosome.genes[r1], chromosome.genes[r2] = chromosome.genes[r2], chromosome.genes[r1]


def crossover(chromosome1, chromosome2):
    end = randint(0, len(chromosome1.genes))
    start = randint(0, end)
    section = chromosome1.genes[start:end]
    offspring_genes = list(gene if gene not in section else None for gene in chromosome2.genes)
    g = (x for x in section)
    for i, x in enumerate(offspring_genes):
        if x is None:
            offspring_genes[i] = next(g)
    offspring = Chromosome(offspring_genes, shuffled=True)

    return offspring
```

GeneticRoutines.py

```
class Route:
    def __init__(self, items, shuffled=False):
        self._genes = items[:]
        if not shuffled:
            shuffle(self._genes)
        self._fitness = None
        self.chance = None

    @property
    def fitness(self):
        distances = (Itemloc.distance(self._genes[i], self._genes[i + 1]) for i in range(-1, len(self._genes) - 1))
        self._fitness = 10 / sum(distances)
        return self._fitness

    @property
    def raw_fitness(self):
        return self._fitness

    @property
    def genes(self):
        return self._genes

    def copy(self):
        clone = Route(self._genes, shuffled=True)
        clone._fitness = self.raw_fitness
        clone.chance = None
        return clone

    def __repr__(self):
        return f'<Route ({self._fitness})>'

    def __str__(self):
        return '->'.join(repr(item) for item in self._genes)
```

TSP.py - C:\Users\kumar\Desktop\project\Genetic-Algorithm-TSP-Picking\TSP.py (3.6.2)

File Edit Format Run Options Window Help

```
from math import hypot
from random import shuffle

class Itemloc:
    def __init__(self, name: str, coords: tuple):
        self._name = name
        self._x, self._y = coords

    @property
    def name(self):
        return self._name

    @property
    def x(self):
        return self._x

    @property
    def y(self):
        return self._y

    @staticmethod
    def distance(item1, item2):
        return hypot(item2.x - item1.x, item2.y - item1.y)

    def __repr__(self):
        return f'{self._name}'


class Item:
    def __init__(self):
        self._items = []

    def add(self, x):
        if isinstance(x, list):
            self.items.extend(x)
        elif isinstance(x, Itemloc):
            self.items.append(x)
        else:
            assert False, "Wrong type"

    @property
    def items(self):
        return self._items
```

TSP.py

GeneticAlgorithm.py - C:\Users\kumar\Desktop\project\Genetic-Algorithm-TSP-Picking\libs\GeneticAlgorithm.py (3.6.2)

File Edit Format Run Options Window Help

```
from libs.GeneticRoutines import selection, mutation, crossover
import time

class GeneticAlgorithm:
    def __init__(self, size, mutation_rate=0.01, ptype=None, args=tuple()):
        assert ptype is not None, 'Population type cannot be None'
        assert type(args) == tuple, 'Arguments must be a tuple instead of ' + str(type(args))
        self._population = [ptype(*args) for _ in range(size)]
        self._mutation_rate = mutation_rate
        self._generation = 0
        self._fittest = self._population[0]
        self.evaluation()

    def individuals(self):
        for chromosome in self._population:
            yield chromosome

    def evaluation(self):
        fitness_sum = sum(chromosome.fitness for chromosome in self._population)
        for chromosome in self._population:
            chromosome.chance = chromosome.fitness / fitness_sum

    def best(self):
        return max(self._population, key=lambda k: k.fitness)

    @property
    def alltime_best(self):
        return self._fittest

    @property
    def generation(self):
        return self._generation

    def next_generation(self):
        new_population = []
        for _ in range(len(self._population)):
            chromosome1 = selection(self._population)
            chromosome2 = selection(self._population)
            new_population.append(crossover(chromosome1, chromosome2))
            mutation(new_population, self._mutation_rate)
        self._population = new_population
        self.evaluation()

    def run(self, seconds=5, reps=None):
        if reps is not None:
            assert isinstance(reps, int), 'Argument `reps` must be of integer type'

            for _ in range(reps - 1):
                pretender = self.best()
                if pretender.fitness > self._fittest.raw_fitness:
                    self._fittest = pretender.copy()

                self._generation += 1
                self.next_generation()
            pretender = self.best()
            if pretender.fitness > self._fittest.raw_fitness:
                self._fittest = pretender.copy()
            self._generation += 1
        else:
            t0 = time.time()
            while True:
                pretender = self.best()
                if pretender.fitness > self._fittest.raw_fitness:
                    self._fittest = pretender.copy()

                self._generation += 1
                if time.time() - t0 >= seconds:
                    break
                self.next_generation()
```

GeneticAlgorithm.py

mod-Graphic.py - C:\Users\kumar\Desktop\project\modified Project\Genetic-Algorithm-TSP-Picking\mod-Graphic.py (3.6.2)

File Edit Format Run Options Window Help

```
from tkinter import *
from TSP import *
from libs.GeneticAlgorithm import GeneticAlgorithm
import pygame
import matplotlib.pyplot as plt
import numpy as np

gen = []
fitn = []
List = []
varList = []
myApp = Tk()
myApp.title("Items-picking list")
myApp.geometry("1000x800")
d = {'Watch':(50,35),'Television':(37,39),'Bag':(26,35),'Book':(14,6),'Mobile':(7,18),'Laptop':(27,15),
     'Refrigerator':(9,12),'Air Conditioner':(42,36),'Pendrives':(16,8),'HDD':(31,37),'Flash card':(45,20),
     'Bottles':(35,8),'Fan':(9,24),'Ear phones':(5,24),'Electric Bulbs':(8,8),'Deodrants':(33,26),'Vaccum Cleaner':(32,40),
     'Sandals':(43,25),'Cameras':(29,24),'Power Banks':(46,27),'Blankets':(25,37),'Hair Dryer':(18,26),'Trimmer':(22,24),
     'Soaps':(18,20),'Washing Machine':(29,35),'Shoes':(20,41),'Goggles':(30,42),'Kerchiefs':(35,42),'Cosmetics':(44,21),
     'Shirts':(15,17),'Pants':(12,23),'Chairs':(23,32)}
p=[]

#myApp.configure(background='')


def map_items_onto_screen(items):
    for item in items:
        y = -int(15 * (item.x - 54))
        x = int(20 * (item.y - 3.5))
        yield (x, y)


def text_labels(items, population_size, mutation_rate):
    global arial_norm, arial_small
    arial_norm = pygame.font.SysFont('arial', 25)
    arial_small = pygame.font.SysFont('arial', 16)
    labels = []
    for item, (posx, posy) in zip(items, map_items_onto_screen(items)):
        labels.append((arial_norm.render(item.name, 1, (255, 255, 255)), (posx - 45, posy - 15)))
    labels.append((arial_small.render('Population size: {}'.format(population_size), 1, (255, 255, 255)), (1100, 10)))
    labels.append((arial_small.render('item count: {}'.format(len(items)), 1, (255, 255, 255)), (1100, 25)))
    labels.append((arial_small.render('Mutation rate: {}'.format(mutation_rate), 1, (255, 255, 255)), (1100, 40)))
    return labels
```

```
    ''' Main loop '''
    while refresh:
        ''' Event handling '''
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                refresh = False
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_SPACE:
                    pause = True
                    while pause:
                        event2 = pygame.event.wait()
                        if event2.type == pygame.KEYDOWN and event2.key == pygame.K_SPACE:
                            pause = False
                        elif event2.type == pygame.QUIT:
                            pause = False
                            refresh = False

        ''' Genetic Algorithm process + statistics '''
        ga.run(reps=skipped_frames)
        current_best = ga.best()
        alltime_fittest = ga.alltime_best
        alltime_fitness = alltime_fittest.raw_fitness


        ''' Drawing part '''
        screen.fill((0, 0, 0))
        for point in map_items_onto_screen(WareHouse.items):
            pygame.draw.circle(screen, (255, 255, 255), point, 3)
        pygame.draw.aalines(screen, (100, 100, 25), True, list(map_items_onto_screen(current_best.genes)))
        pygame.draw.aalines(screen, (255, 255, 255), True, list(map_items_onto_screen(alltime_fittest.genes)))
                #gen.append(ga.generation)
                #fit.append(current_best.raw_fitness)
        x=ga.generation
        y=current_best.raw_fitness
        gen.append(x)
        fitn.append(y)
        screen.blit(arial_small.render('Generation: {}'.format(x), 1, (255, 255, 255)), (1100, 55))
        screen.blit(arial_small.render('Fitness: {:.4f}'.format(alltime_fitness), 1, (255, 255, 255)), (1100, 70))
        screen.blit(arial_small.render('Current Fitness: {:.4f}'.format(y), 1, (255, 255, 0)), (1100, 85))
        for label in stat_labels:
            screen.blit(label[0], label[1])
        pygame.display.flip()
```

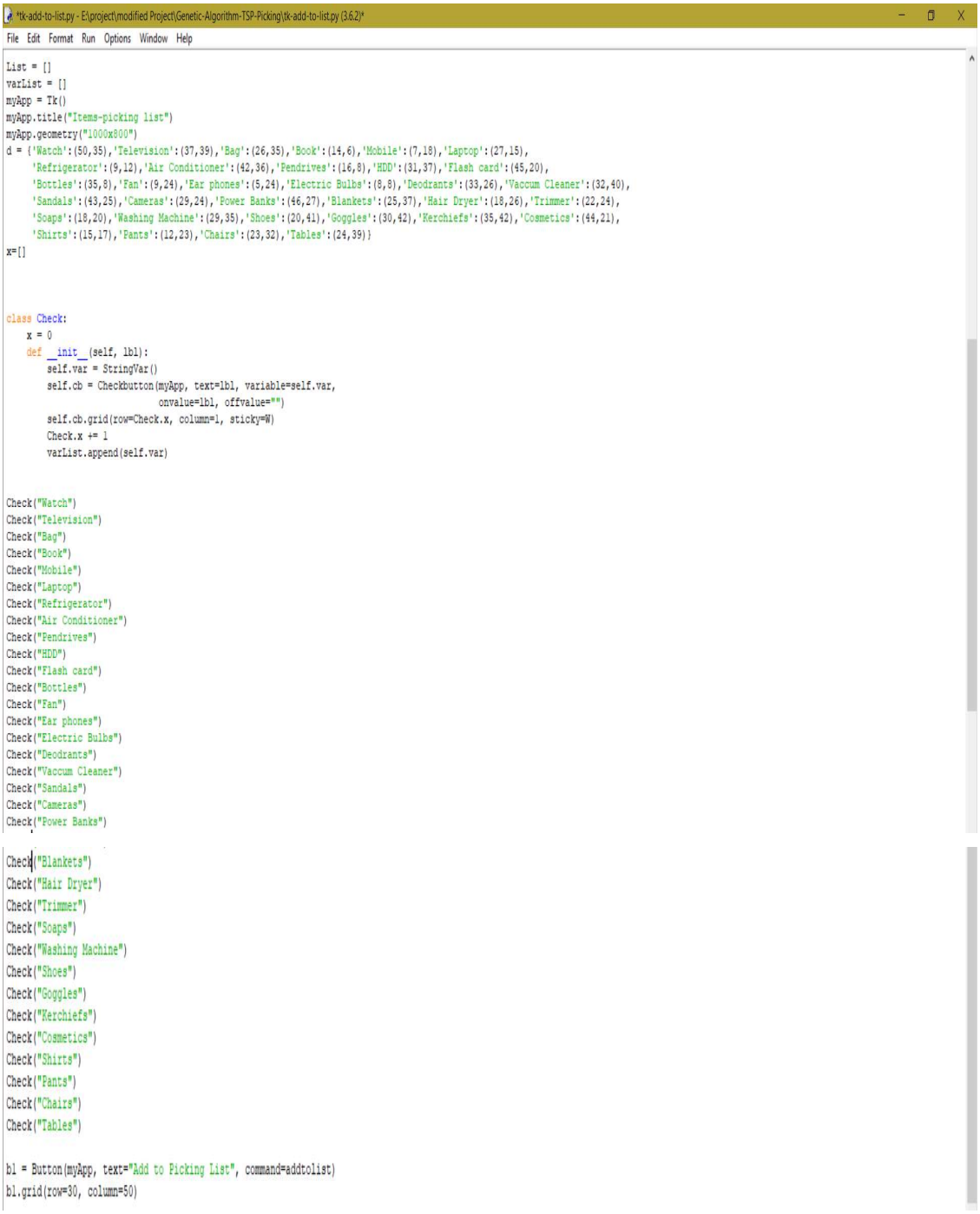
```
def addtolist():
    global List

    List = []
    for item in varList:
        if item.get() != "":
            List.append(item.get())

    for i in List:
        p.append(d[i])

    s=[]
    l=List
    for i in range(len(l)):
        s.append(Itemloc(l[i],p[i]))
    WareHouse = Item()

    WareHouse.add(s)

    population_size = 20
    mutation_rate = 0.01
    skipped_frames = 108

    pygame.init()
    pygame.display.set_caption('Picking Routes in warehouse')
    screen = pygame.display.set_mode((1300, 780))
    stat_labels = text_labels(WareHouse.items, population_size, mutation_rate)
    ga = GeneticAlgorithm(population_size, mutation_rate, ptype=Route, args=(WareHouse.items,))
    alltime_fittest = ga.alltime_best
    alltime_fitness = 0
    refresh = True
```

```
    print('Selected Items:', end=' ')
    print(*(item for item in WareHouse.items), sep=', ')
    print('Best route:', alltime_fittest)
    print('Best fitness:', alltime_fitness)
    plt.plot(gen,fitn)
    plt.xlabel('Generations')
    plt.ylabel('Fitness')
    plt.show()

class Check:
    x = 0
    def __init__(self, lbl):
        self.var = StringVar()
        self.cb = Checkbutton(myApp, text=lbl, variable=self.var,
                              onvalue=lbl, offvalue="")
        self.cb.grid(row=Check.x, column=1, sticky=W)
        Check.x += 1
        varList.append(self.var)


Check("Watch")
Check("Television")
Check("Bag")
Check("Book")
Check("Mobile")
Check("Laptop")
Check("Refrigerator")
Check("Air Conditioner")
Check("Pendrives")
Check("HDD")
Check("Flash card")
Check("Bottles")
Check("Fan")
Check("Ear phones")
Check("Electric Bulbs")
Check("Deodrants")
Check("Vaccum Cleaner")
Check("Sandals")
Check("Cameras")
Check("Power Banks")
Check("Blankets")
Check("Hair Dryer")
Check("Trimmer")
Check("Soaps")
Check("Washing Machine")
Check("Shoes")
```

```
Check("Goggles")
Check("Kerchiefs")
Check("Cosmetics")
Check("Shirts")
Check("Pants")
Check("Chairs")


bl = Button(myApp, text="Add to Picking List", command=addtolist)
bl.grid(row=30, column=50)
```

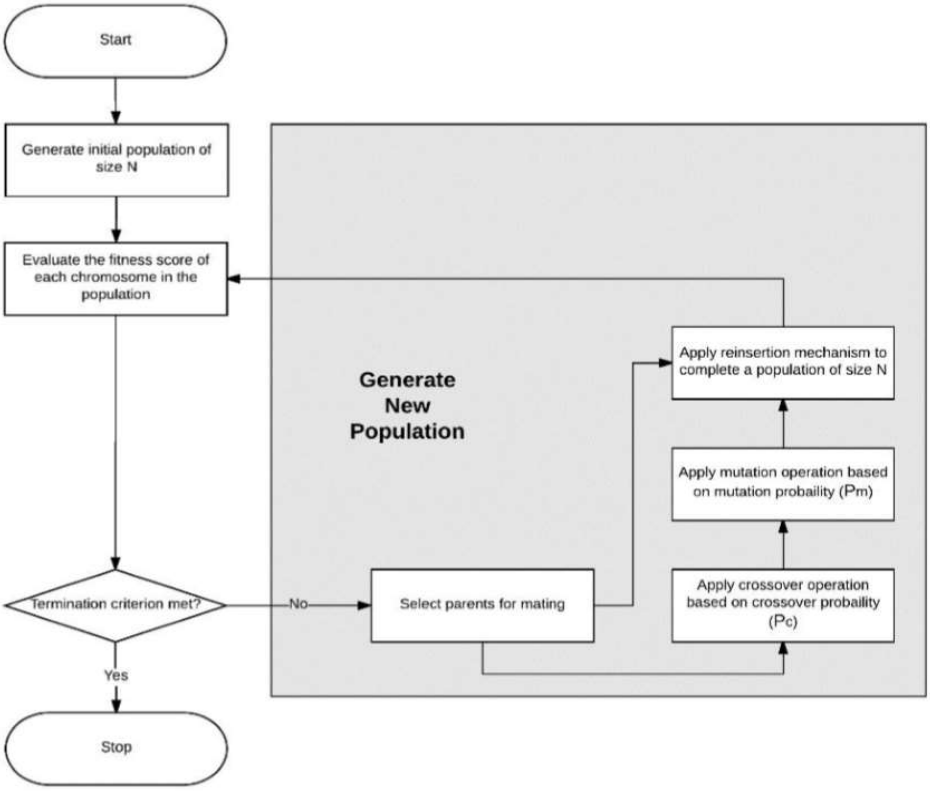
mod-Graphic.py


tk-add-to-list.py - E:\project\modified Project\Genetic-Algorithm-TSP-Picking\tk-add-to-list.py (3.6.2)
File Edit Format Run Options Window Help
from tkinter import *
from TSP import *
from libs.GeneticAlgorithm import GeneticAlgorithm
def addtolist():
global List
List = []
for item in varList:
if item.get() != "":
List.append(item.get())
for i in List:
x.append(d[i])
s=[]
l=List
for i in range(len(l)):
s.append(Itemloc(l[i],x[i]))
WareHouse = Item()
WareHouse.add(s)
print('Selected Items:', end=' ')
print(*(item for item in WareHouse.items), sep=', ')
ga = GeneticAlgorithm(100, mutation_rate=0.5, ptype=Route, args=(WareHouse.items,))
ga.run(seconds=40)
fittest = ga.alltime_best
best_fitness = fittest.fitness
print('Best route:', fittest)
print('Best fitness:', best_fitness)
print('Generations:', ga.generation)
#print(s)

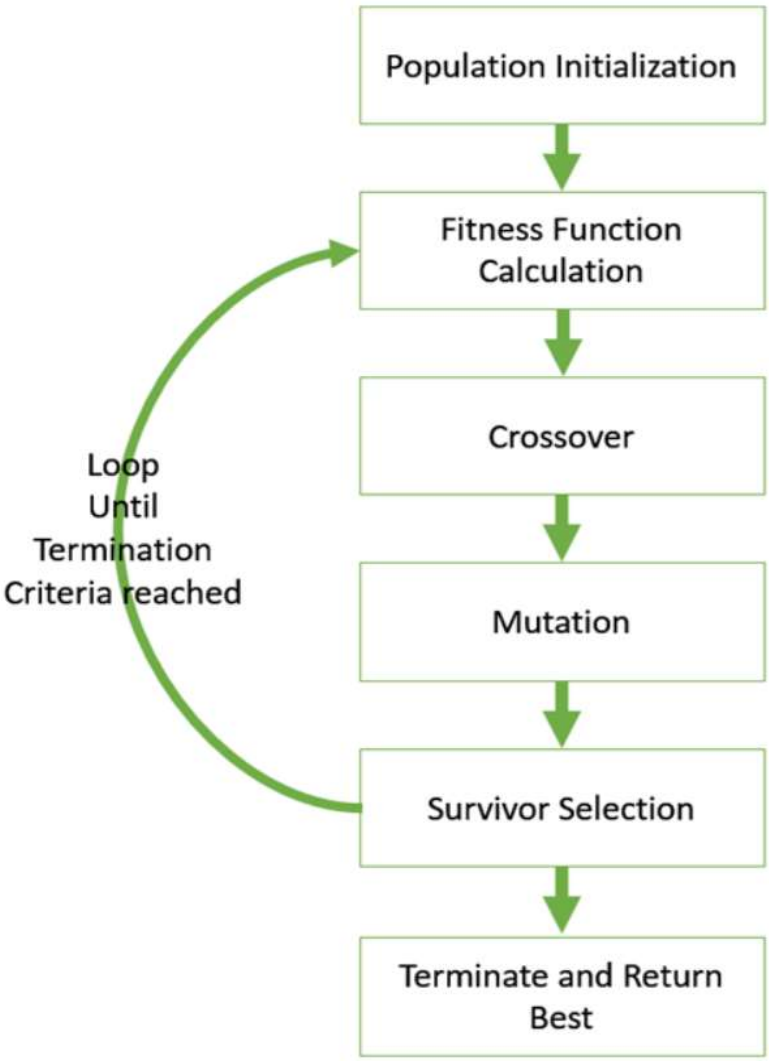


tk-add-to-list.py

LIST OF FIGURES



Algorithm Flow graph



Generalization of Algorithm

REFERENCES

1. De Koster, R., Le-Duc, T. and Roodbergen, K., "Design and Control of Warehouse Order Picking: A Literature Review", European Journal of Operational Research, Vol. 182, No. 2, pp.481-501, 2007.

2. Dallari, F., Marchet, G. and Melacini, M., "Design of Order Picking System", The International Journal of Advanced Manufacturing Technology, Vol. 42, No. 1-2, pp.112, 2008.

3. Tuna G., Tunçel G., "Depo Yönetiminde Sipariş Toplama Sistemleri: Bir Literatür Araştırması", Dokuz Eylül Üniversitesi Mühendislik Fakültesi, Mühendislik Bilimleri Dergisi, Vol. 14, No. 42, pp.15-31, 2012.

4. Roodbergen, K. and De Koster, R., "Routing Order Pickers in a Warehouse with a Middle Aisle", European Journal of Operational Research, Vol. 133, No. 1, pp.32-43, 2001.

5. Roodbergen, K. and De Koster, R., "Routing Methods for Warehouses with Multiple Cross Aisles", International Journal of Production Research, Vol. 39, No. 9, pp.18651883, 2001.

6. De Koster, R. and Poort, E., "Routing Orderpickers in a Warehouse: A Comparison Between Optimal and Heuristic Solutions", IIE Transactions, Vol. 30, No. 5, pp.469480, 1998.

7. Roodbergen K.J., "Storage Assignment for Order Picking in Multiple-Block Warehouses", in Riccardo Manzini ed., Warehousing in the Global Supply Chain, Advanced Models, Tools and Applications for Storage Systems, pp: 139-155, Springer, London, 2012.

8. Park B.C., "Order Picking: Issues, Systems and Models", in Riccardo Manzini ed., Warehousing in the Global Supply Chain, Advanced Models, Tools and Applications for Storage Systems, pp: 1-30, Springer, London, 2012.

9. Van den Berg, J., "A Literature Survey on Planning and Control of Warehousing Systems", IIE Transactions, Vol. 31, No. 8, pp.751-762, 1999.

10. Rouwenhorst, B., Reuter, B., Stockrahm, V., van Houtum, G., Mantel, R. and Zijm, W., "Warehouse Design and Control: Framework and Literature Review", European Journal of Operational Research, Vol. 122, No. 3, pp.515-533, 2000.

11. Gu, J., Goetschalckx, M. and McGinnis, L., "Research on Warehouse Operation: A Comprehensive Review", European Journal of Operational Research, Vol. 177, No. 1, pp.1-21, 2007.

12. Petersen, C. and Aase, G., "A Comparison of Picking, Storage, and Routing Policies in Manual Order Picking", International Journal of Production Economics, Vol. 92, No. 1, pp.11-19, 2004.

13. Won, J. and Olafsson, S., "Joint Order Batching and Order Picking in Warehouse Operations", International Journal of Production Research, Vol. 43, No. 7, pp.14271442, 2005.

14. Daniels, R., Rummel, J. and Schantz, R., "A Model for Warehouse Order Picking", European Journal of Operational Research, Vol. 105, No. 1, pp.1-17, 1998.

15. Ficko M., Klancnik S., Brezovnik S., Balic J., Brezocnik M., Lerher T., "Intelligent Optimization Methods for Industrial Storage Systems", in Riccardo Manzini ed., Warehousing in the Global Supply Chain, Advanced Models, Tools and Applications for Storage Systems, pp: 341-370, Springer, London, 2012.

16. Chen, Z., Xie, S. and Wu, D., "Order Picking Path Optimization Based on Genetic Algorithm", Key Engineering Materials, Vol. 464, pp.379-382, 2011.

17. Hsu, C., Chen, K. and Chen, M., "Batching Orders in Warehouses by Minimizing Travel Distance with Genetic Algorithms", Computers in Industry, Vol. 56, No. 2, pp.169-178, 2005.

18. Tsai, C., Liou, J. and Huang, T., "Using a Multiple-GA Method to Solve the Batch Picking Problem: Considering Travel Distance and Order Due Time", International Journal of Production Research, Vol. 46, No. 22, pp.6533-6555, 2008.

19. Öncan, T., "A Genetic Algorithm for the Order Batching Problem in Low-Level Pickerto-Part Warehouse Systems", Proceedings of the International MultiConference of Engineers and Computer Scientists 2013, IMECS 2013, Hong Kong, 13-15 March 2013, Hong Kong, 2013.

20. Bottani, E., Cecconi, M., Vignali, G. and Montanari, R., "Optimisation of Storage Allocation in Order Picking Operations through a Genetic Algorithm", International Journal of Logistics Research and Applications, Vol. 15, No. 2, pp.127-146, 2012.

21. Molnár, B. and Lipovszki, G., "Multi-Objective Routing and Scheduling of Order Pickers in a Warehouse", International Journal of Simulation, Vol. 6, No. 5, pp.22-32, 2005.

22. Seyedrezaei, M., Najafi, S.E., Aghajani, A. and Valami, H.B., "Designing a Genetic Algorithm to Optimize Fulfilled Orders in Order Picking Planning Problem with Probabilistic Demand", International Journal of Research in Industrial Engineering, Vol. 1, No. 2, pp.40-57, 2012.

23. Khojasteh-Ghamari, Y. and Son, J.D., "Order Picking Problem in a Multi-Aisle Automated Warehouse Served by a Single Storage/Retrieval Machine", International Journal of Information and Management Sciences, Vol. 19, No. 4, pp.651-665, 2008.

24. Cheng, R., Gen, M. and Sasaki, M., "Film-copy Deliverer Problem Using Genetic Algorithms", Computers & Industrial Engineering, Vol. 29, No. 1-4, pp.549-553, 1995.

25. Şahin Y., Eroğlu A., "Sipariş Toplama ve Kapasite Kısıtlı Araç Rotalama Problemlerinin Hiyerarşik Çözümü", Süleyman Demirel Üniversitesi Mühendislik Bilimleri ve Tasarım Dergisi, Vol. 3, No. 1, pp.15-28, 2015.

26. Şahin Y., Kulak O., "Depo Operasyonlarının Planlanması için Genetik Algoritma Esaslı Modeller", Uluslararası Alanya İşletme Fakültesi Dergisi, Vol. 5, No. 3, pp.141-153, 2013.

project document

ORIGINALITY REPORT

30%	23%	15%	22%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Siddaganga Institute of Technology Student Paper	4%
2	Submitted to University of Nottingham Student Paper	3%
3	Submitted to Universiti Tenaga Nasional Student Paper	2%
4	ijarcsse.com Internet Source	2%
5	www.worldsbestinformation.org Internet Source	1%
6	link.springer.com Internet Source	1%
7	aakashtechsupportdocs.readthedocs.org Internet Source	1%
8	www.inventoryops.com Internet Source	1%
9	holisollogistics.com	

	Internet Source	1%
10	fr.scribd.com Internet Source	1%
11	Submitted to Universiti Teknologi Malaysia Student Paper	1%
12	www.camcode.com Internet Source	1%
13	www.professionalcipher.com Internet Source	1%
14	wiki.syncleus.com Internet Source	1%
15	www.sihs.com Internet Source	1%
16	Submitted to Cranfield University Student Paper	1%
17	Kees Jan Roodbergen, René de Koster. "Routing order pickers in a warehouse with a middle aisle", European Journal of Operational Research, 2001 Publication	1%
18	"Warehousing in the Global Supply Chain", Springer Nature, 2012 Publication	1%

19	www.conveyco.com Internet Source	<1%
20	cerasis.com Internet Source	<1%
21	"Proceedings of the Tenth International Conference on Management Science and Engineering Management", Springer Nature, 2017 Publication	<1%
22	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
23	www.riejournal.com Internet Source	<1%
24	simulation.su Internet Source	<1%
25	www.tutorialspoint.com Internet Source	<1%
26	Submitted to Champlain College Student Paper	<1%
27	J. Won. "Joint order batching and order picking in warehouse operations", International Journal of Production Research, 4/1/2005 Publication	<1%
	www.springerlink.com	

28	Internet Source	<1%
29	Submitted to Kolej Universiti Linton Student Paper	<1%
30	Emin Erkan Korkmaz. "Multi-objective Genetic Algorithms for grouping problems", Applied Intelligence, 12/31/2008 Publication	<1%
31	Submitted to Wright State University Student Paper	<1%
32	shop.tarjomeplus.com Internet Source	<1%
33	Stefan Vonolfen, Monika Kofler, Andreas Beham, Michael Affenzeller, Werner Achleitner. "Optimizing assembly line supply by integrating warehouse picking and forklift routing using simulation", Proceedings Title: Proceedings of the 2012 Winter Simulation Conference (WSC), 2012 Publication	<1%
34	baadalsg.inflibnet.ac.in Internet Source	<1%
35	tigerprints.clemson.edu Internet Source	<1%
36	Pan, J.C.H.. "Evaluation of the throughput of a	

	multiple-picker order picking system with congestion consideration", Computers & Industrial Engineering, 200809 Publication	<1%
37	repub.eur.nl Internet Source	<1%
38	scholar.uwindsor.ca Internet Source	<1%
39	fr.slideshare.net Internet Source	<1%
40	Xi Xiang, Changchun Liu, Lixin Miao. "Storage assignment and order batching problem in Kiva mobile fulfilment system", Engineering Optimization, 2018 Publication	<1%
41	d-nb.info Internet Source	<1%
42	www.emeraldinsight.com Internet Source	<1%
43	www.tandfonline.com Internet Source	<1%
44	www.thinkmind.org Internet Source	<1%
45	Osman Kulak. "Joint order batching and picker routing in single and multiple-cross-aisle	<1%

warehouses using cluster-based tabu search algorithms", Flexible Services and Manufacturing Journal, 06/19/2011
Publication

---

46 Limere, Veronique, Melih Celik, Aditya Pradhan, and Mallory Soldner. "Warehousing efficiency in a small warehouse", 2011 IEEE Workshop On Computational Intelligence In Production And Logistics Systems (CIPLS), 2011.
Publication

<1%

---

47 B. Rouwenhorst, B. Reuter, V. Stockrahm, G.J. van Houtum, R.J. Mantel, W.H.M. Zijm. "Warehouse design and control: Framework and literature review", European Journal of Operational Research, 2000
Publication

<1%

---

48 "Computational Science and Its Applications – ICCSA 2007", Springer Nature, 2007
Publication

<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		