Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 159 (2019) 261–270

Procedia Computer Science

www.elsevier.com/locate/procedia

23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

A quantum genetic algorithm for pickup and delivery problems with coalition formation

Yara Rizk[a], Mariette Awad[a,*]

[a]*Department of Electrical and Computer Engineering, American University of Beirut, Beirut, Lebanon*

Abstract

With the “last mile” of the delivery process being the most expensive phase, autonomous package delivery systems are gaining traction as they aim for faster and cheaper delivery of goods to city, urban and rural destinations. This interest is further fueled by the emergence of e-commerce, where many applications can benefit from autonomous package delivery solutions. However, the environment stochasticity, variability and task complexity for autonomous operation make it difficult to deploy such systems in real-world applications without the incorporation of advanced machine learning and optimization algorithms. Moving away from designing a “one size fits all” agent to solve the outdoor package delivery problem and considering ad-hoc teams of agents trained within a data-driven framework could provide the answer. In this work, we propose a delivery scheduling algorithm for heterogeneous multi-agent systems using the pickup and delivery problem (PDP) formulation. Specifically, a 3-index mixed integer program-based PDP that allows coalition formation (PDP-CF) among agents is derived to allow multi-agent PDP schedules. We propose a quantum genetic algorithm to solve for the schedule since it is can better handle the large computational complexity of PDP-CF. Multiple PDP scenario simulations show the merits of the proposed approach.

Keywords: Pickup and delivery problems; Coalition formation; Quantum genetic algorithm, Cooperating agents

1. Introduction

With the automation of many everyday tasks resulting from cheaper and better electronics and robotic systems, some tasks such as package delivery still lag behind due to many difficulties. The emergence of Internet of Things and smart cities can help make autonomous delivery commonplace, benefiting many industries like retail, healthcare and emergency response. As e-commerce becomes more popular, evidenced by its $1.471 trillion sales worldwide in 2015 [22], shipping products to consumers is taking up a larger portion of companies’ revenues. The “last mile” of the delivery process is the most expensive phase, with Amazon reportedly spending approximately $11.5 billion in 2015

[*] Corresponding author. Tel.: +961-1-350000 ext. 3528 ; fax: +961-1-744462.
E-mail address: mariette.awad@aub.edu.lb

on shipping costs [19]. Furthermore, the number of packages shipped by Amazon is estimated at about 608 million packages a year and is steadily increasing [23]. With consumers expecting faster and cheaper delivery, companies like Amazon are looking for better package delivery systems; autonomous package delivery systems may be the answer. However, widespread deployment has yet to happen due to the complexity of real-world environments.

The autonomous package delivery problem consists of multiple sub-problems from different fields including scheduling, object transport in robotics, cooperative robot navigation, and others. While pickup and delivery problems (PDPs) have been extensively researched in the field of scheduling [18, 17] and to a certain extent in conjunction with robotics [5], packages are generally assumed to be transported by one robot. The most basic PDP formulation schedules the delivery of packages by modeling the problem as a graph and finds the routes that minimize a cost function. However, allowing multi-robot systems to deliver a single package would increase the number of feasible assignments, especially when capacity constraints are considered.

To date, PDP formulations have assumed that vehicles operate independently of other vehicles. In this work, we assume that vehicles are allowed to cooperate when transporting a package. Specifically, we consider a PDP formulation with coalition formation (CF), termed PDP-CF, where multiple vehicles can form a coalition to deliver a package that may exceed their individual payloads. A coalition is defined to be a group of robots cooperatively transporting one or more packages simultaneously. Specifically, we develop a 3-index mixed integer program (MIP) formulation that solves for a delivery schedule that minimizes a certain cost function. This leads to an exponentially larger search space for PDP but could lead to more cost efficient solutions. Due to the larger search space, we propose a quantum genetic algorithm (QGA) that could traverse this space more efficiently and converge to a better solution. QGA's chromosomes would encode the 3-index MIP optimization variables. Empirically, the proposed algorithm and solver achieve cost effective schedules when solving static PDP. Results also showed that imposing capacity and overlap constraints was challenging since the encoding does not prevent QGA from exploring the infeasible region; this hindered QGA's ability to converge to a solution in a reasonable time.

2. Related Work

2.1. PDP Formulations

A PDP aims to find a schedule that allows a set of vehicles (or robots) to execute transportation requests such as delivering a set of packages from source to destination. The goal is to find a schedule that satisfies various constraints such as capacity, time window and priority, while minimizing an objective function such as distance traveled, energy consumed, or delivery time. A general PDP formulation was proposed in 1995 [21] and many variants with different constraints have been studied. PDPs were shown to be NP-hard by reducing them to a traveling salesman problem [12], which implies that finding an optimal schedule in polynomial time may not be feasible. Some of the main assumptions in a PDP formulation are:

- **Number of vehicles** that will deliver items. Single vehicle and multi-vehicle PDPs have been studied.
- **Vehicle start and end locations** specify the start location(s) (e.g. warehouse) and return location(s), if any.
- **Item pickup and delivery locations** specify whether items are picked up from a single location or from different locations and whether they will be delivered to a single location or individual locations.
- **Capacities and demands** specify a vehicle's maximum capacity and an item's demand or payload.
- **Time windows** specify the earliest and latest times an item can be picked up and delivered, respectively.
- **Maximum route durations** specify the amount of time a vehicle may spend on any given delivery route.
- **Maximum transport times** specify the amount of time an item can spend in a vehicle before it is delivered.
- **Transfers or transshipments** specify whether items can be exchanged among vehicles at intermediary locations between pickup and delivery.

Static PDPs [3] assume that all delivery requests and vehicles are known before a schedule is formed, while dynamic PDPs [4] allow the addition of new requests and vehicles as the delivery schedule is being formed. Dynamic PDP further complicates PDPs since not all the information is available from the start. Thus, greedy algorithms that reach suboptimal solutions tend to be adopted. Online PDP algorithms solve the PDP algorithm in an online fashion as vehicles execute delivery tasks [6].

PDPs have been most commonly formulated as constrained optimization problems with both continuous and integer variables. The objective function may consist of multiple criteria including minimizing distance traveled, total duration, completion time, client inconvenience, and delivery cost, to name a few. Constraints are derived from the initial assumptions of the formulation (e.g. time windows, capacities) but also include constraints that ensure the validity of a schedule such as delivering an item after it has been picked up. Compact formulations have been proposed to enable more efficient solutions using general purpose optimization solvers. A 2-index formulation was proposed in [15] based on solving a PDP with time windows (PDPTW) as a Hamiltonian tour problem. However, this approach did not perform well on large scale PDPs. Another 2-index formulation was derived based on the vehicle routing problem with time windows formulation and explicitly assigning vehicle routes [10].

2.2. PDP Solvers

Many heuristics and meta-heuristics have been proposed to find delivery schedules using search-based algorithms. From optimization, branch-and-cut or branch-and-bound MIP algorithms have been commonly adopted to solve PDP variants including PDPTW and transfers [7]. Particle swarm optimization [14], genetic algorithms [1] and simulated annealing [2] have also been adopted in large neighborhood PDPTW. Furthermore, heuristics and meta-heuristics have been developed to efficiently converge to optimal delivery schedules, as in [8]. For example, a 2-phase heuristic was proposed in [11], and a branch-and-cut algorithm for 2-index PDPTW [15].

Focusing on evolutionary algorithms for PDP, Wang et al. [25] formulated a MIP model for simultaneous PDPTW and encoded the optimization variables into GA's chromosomes. To solve PDPTW, [16] proposed a grouping GA which implies that genes represent multiple delivery requests as opposed to one request. Xiao et al. [26] proposed a quantum ant colony optimization algorithm to solve vehicle routing problems. Zhang et al. [27] formulated a hybrid quantum evolutionary algorithm for PDP and demonstrated its superior exploration capabilities compared to GA. Dakroub et al. [9] adopted a GA for intelligent carpooling by implementing multiple population threads with each chromosome representing a driver and each gene encoding passengers with the driver. Crossover and mutation operations were modified to prevent GA from exiting the feasible region. Ursani et al. [24] developed a two-phase optimization solver based on GA for vehicle routing problems. The algorithm breaks down the original optimization problem into smaller ones, optimizes them, and then combines their solutions and fine tunes the results using a de-optimization workflow. GA encodes the order of customers. Finally, Jia et al. [13] encoded target and drone information into GA's chromosomes to assign targets related to flight trajectories to cooperating drones.

3. Methodology

We propose a PDP formulation, hereafter referred to as PDD-CF, which allows the formation of coalitions among different agents. This would increase the number of feasible assignments, especially when capacity constraints are considered. Then, we discuss how QGA can find schedules for PDP-CF. Table 1 summarize the adopted notation.

3.1. Motivating Example

We consider an example to illustrate the benefits of allowing CF in PDP. Given a set of robots and packages, described in Table 2, we need to find the robot-package assignment that will minimize the total distance traveled by all the robots. In this example, we simplify the PDP formulation to only consider the source, destination and mass of the packages, in addition to the payloads of robots. The environment is a simple 3x3 grid, as shown in Figure 1. We define transfers as a package being handed to a different robot at a location other than its final destination. Coalitions are defined as a group of robots collaboratively transporting a package or multiple packages, i.e. the robots share the load of the package(s). If transfers and coalitions are not allowed, the assignment in Table 3 is optimal and leads to a total distance of 12 units traveled by all robots. The best solution is obtained when both coalitions and transfers are allowed: a total distance of 5 units is traveled by the robots, as shown in Table 3. PICKUP(R_1,C) implies that robot R_1 has picked up package C from its source. Similarly, DELIVER(R_1,C) implies that robot R_1 has dropped off package C at its destination. Finally, TRANSFER(R_3,R_1,D) means that robot R_3 has handed off package D to robot R_1.

Table 1. Nomenclature

Symbol	Definition	Symbol	Definition
V	Set of vehicles, $v \in V$	m	Number of vehicles
P	Set of packages, $p \in P$	n	Number of packages
$K_v = 2^V$	Set of coalitions, $k_v = \{v_i\} \in K_v$, $v_i \in V$	G	Graph representing a PDP instance
A	Arc in graph G, $(i, j) \in A$	N	Nodes in graph G
P	Pickup nodes	D	Delivery nodes
x_{ijk}	Boolean optimization variable	c_{ijk}	Arc cost
Q_{jk}	Capacity of coalition k at node j	t_{ij}	Travel time associated with (i, j)
B_{jk}	Time at which coalition k starts servicing node j	M	Constant
q_i	Demand at node i	$[e_i, \ell_i]$	Time window of node i
k_s	Maximum allowable coalition size	d_{ij}	Distance between nodes i and j
d_{ijk}	Distance traveled by a coalition k from node i to j	$\|k\|$	Number of vehicles in coalition k
$cap(v)$	Capacity of a vehicle v	$dem(p)$	Payload demanded by a package p

Table 2. Robot and Package Setup

Robot	Location (x,y)	Payload (units)	Package	Source Location (x,y)	Destination Location (x,y)	Mass (units)
R_1	(0,1)	2	A	(0,1)	(2,1)	3
R_2	(0,1)	2	B	(0,1)	(1,2)	1
R_3	(1,0)	3	C	(1,0)	(2,1)	1
R_4	(1,0)	1	D	(1,0)	(1,2)	1

Table 3. A possible schedule

Transfers and CF not allowed			Transfers and CF allowed		
Time step	Action	Distance traveled	Time step	Action	Distance traveled
1	PICKUP(R_1, C)	2	1	PICKUP(R_1+R_2, A)	0
1	PICKUP(R_2, B)	0	1	PICKUP(R_1+R_2, B)	0
1	PICKUP(R_3, A)	2	1	PICKUP(R_3, C)	0
1	PICKUP(R_4, D)	0	1	PICKUP(R_3, D)	0
2	DELIVER(R_1, C)	2	2	TRANSFER(R_1+R_2, R_3, A)	2
2	DELIVER(R_2, B)	2	2	TRANSFER(R_3, R_1, D)	1
2	DELIVER(R_3, A)	2	3	DELIVER(R_1, B)	1
2	DELIVER(R_4, D)	2	3	DELIVER(R_1, D)	
			3	DELIVER(R_3, A)	1
			3	DELIVER(R_3, C)	
	Total Cost	12		Total Cost	5

3.2. A 3-index MIP PDP-CF Formulation

PDP is most commonly formulated as a 3-index MIP problem [20]; it is represented as a complete graph $G(N, A)$, where $N = \{0, 1, ..., 2n + 1\}$ denotes the set of nodes in the graph and A the set of arcs or connections between the nodes in N. We assume n total transportation requests or packages; with each request, we associate a pickup node $i \in P$ and delivery node $n + i \in D$. To simplify the notation, we define $P = \{1, ..., n\} \subset N$ to be the set of pickup nodes and $D = \{n + 1, ..., 2n\} \subset N$ to be the set of delivery nodes, with nodes $0 \in N$ and $2n + 1 \in N$ representing the origin and final depots. The set of available vehicles is denoted by $V = \{1, 2, ..., v, ..., m\}$ with cardinality $|V| = m$ and capacity cap_v. Unlike many works in the literature, we do not assume V to be homogeneous, i.e. the capabilities of the vehicles in V are diverse. Figure 2 illustrates this graphical representation for $n = 2$.

The objective function in (1) minimizes the total routing cost by searching the space of coalitions instead of the space of vehicles. This leads to an exponentially larger search space since the set of coalitions is the power set of V, $K = 2^V$. Furthermore, c_{ijk} is the cost of a coalition k traversing arc $(i, j) \in A$. Existing formulations assumed a routing cost that is independent of the vehicle traversing the route since homogeneous vehicles were considered. However, this does not apply to our formulation since the size of the coalition influences this cost. The routing cost can be one

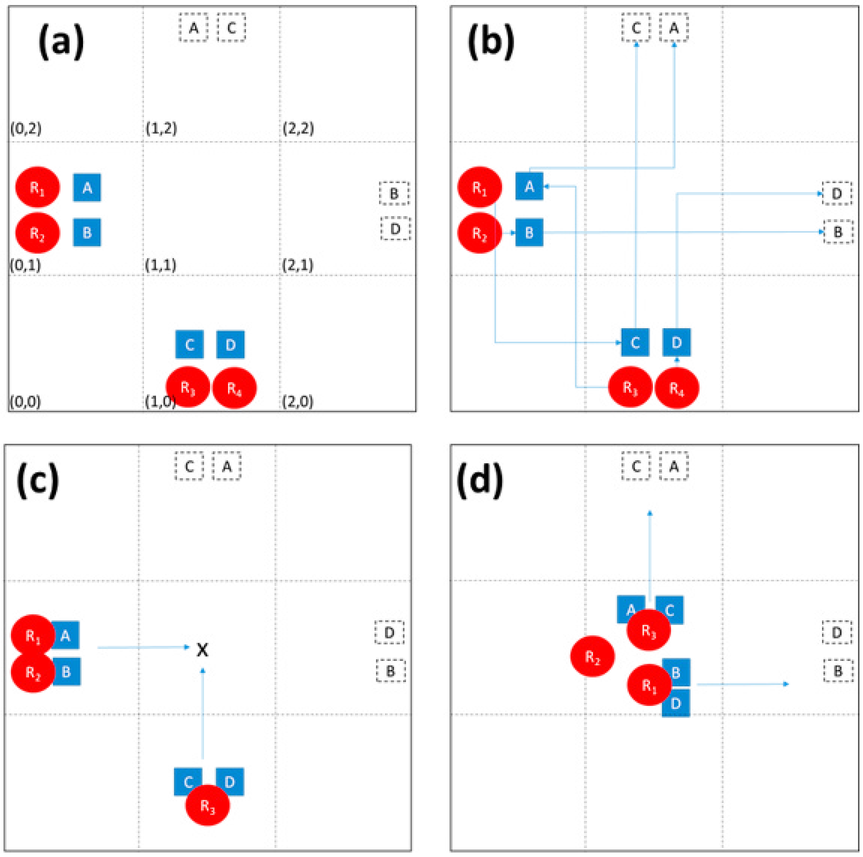


Fig. 1. Illustrative PDP example: (a) Initial map, (b) Solution without transfers or coalitions, (c) Step 1 of solution with transfers and coalitions, (d) Step 2 of solution with transfers and coalitions.

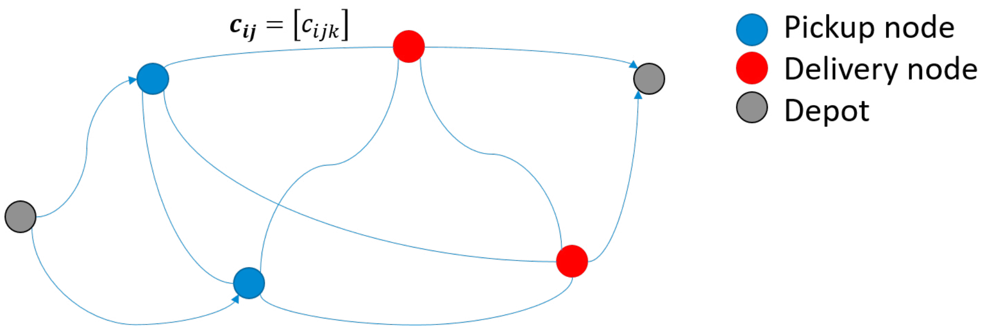


Fig. 2. PDP's Graphical Representation

of many functions such as traveled distance, delivery time, consumed energy or a combination of the three. x_{ijk} is a binary variable that is equal to 1 when coalition $k \in K$ traverses arc $(i, j) \in A$.

$$\min_x \sum_{k\in 2^V} \sum_{i\in N} \sum_{j\in N} c_{ijk}x_{ijk} \tag{1}$$

$$\sum_{k\in K} \sum_{j\in N} x_{ijk} = 1 \quad \forall i \in P \tag{2}$$

$$\sum_{j\in N} x_{ijk} - \sum_{j\in N} x_{n+i,j,k} = 0 \quad \forall i \in P; k \in K \tag{3}$$

$$\sum_{j\in N} x_{0jk} = 1 \quad \forall k \in K \tag{4}$$

$$\sum_{i\in N} x_{i,2n+1,k} = 1 \quad \forall k \in K \tag{5}$$

$$\sum_{j\in N} x_{jik} - \sum_{j\in N} x_{ijk} = 0 \quad \forall i \in P \cup D; k \in K \tag{6}$$

$$Q_{jk} \geq Q_{ik} + q_j - M(1 - x_{ijk}) \quad \forall i \in N; j \in N; k \in K \tag{7}$$

$$B_{jk} \geq B_{ik} + t_{ij} - M(1 - x_{ijk}) \quad \forall i \in N; j \in N; k \in K \tag{8}$$

$$B_{ik} + t_{i,n+i} \leq B_{n+i,k} \quad \forall i \in P; k \in K \tag{9}$$

$$e_i \leq B_{ik} \leq \ell_i \quad \forall i \in N; k \in K \tag{10}$$

$$\max\{0, q_i\} \leq Q_{ik} \leq \min\{cap_k, cap_k + q_i\} \quad \forall i \in N; k \in K \tag{11}$$

$$cap_k = \sum_{v\in k} cap_v \tag{12}$$

$$x_{ijk} \in \{0, 1\} \quad \forall i \in N; j \in N; k \in K \tag{13}$$

$$x_{ijk} + x_{ijk'} = 1 \forall k \cap k' \neq \emptyset \tag{14}$$

We impose multiple constraints on the optimization problem to obtain a valid delivery schedule. First, we ensure that a transportation request is assigned to only one coalition in (2) and the corresponding pickup and delivery nodes are visited by this same coalition in (3). Then, we ensure that each route starts at the origin depot and terminates at the final depot by adding the constraints in (4) and (5). The constraint in (6) guarantees that each coalition entering a node will leave this node. Capacity and time constraints are imposed by (7) and (8), where Q_{jk} is the capacity of coalition k at node j, t_{ij} is the travel time associated with arc (i, j), B_{jk} is time at which coalition k starts servicing node j, and M is a sufficiently large constant. t_{ij} includes the service time of each node and obeys the triangle inequality. $q_i \geq 0$ is the load associated with a pickup node $i \in P$; $q_{n+i} = -q_i$ is the load associated with a delivery node $n + i \in D$.

In PDP, we need to ensure that a pickup node is visited before a delivery node by adding the constraint in (9). Since time windows $[e_i, \ell_i]$ are associated with each node $i \in N$, the constraint in (10) ensures the schedule does not allow coalition k to visit a node i before e_i or after ℓ_i. To ensure the maximum capacity of each coalition is never exceeded, the constraint in (11) is imposed. The capacity of a coalition is defined as the sum of the capacities of the vehicles in the coalition, as shown in (12). Finally, the integrality of variables is satisfied by adding the constraint in (13).

Since we are considering coalitions of cooperating vehicles instead of independent vehicles, we add a set of constraints that ensures coalitions do not overlap, as shown in (14). If coalitions k and k' contain common vehicles (their intersection is not empty), then only one of these coalitions can be used. This set of constraints is proportional to the size of the set of coalitions and equal to $|K|^2/2 = 2^{2|V|-1}$. To reduce the size of the search space and number of constraints, we limit the maximum size of a coalition. Instead of considering all subsets of V, we only consider all subsets with a maximum cardinality $k_s << m$.

Therefore, three main modifications to the 3-index MIP in [20] have been proposed in this work: 1) the search space of possible solutions is the power set of vehicles instead of the set of vehicles, 2) the cost function depends on the coalitions, i.e c_{ij} is replaced by c_{ijk}, and 3) a set of constraints was added to ensure that coalitions do not overlap.

Table 4 summarizes the computational complexity of 3-index MIP formulation with and without CF. The graph size is based on the number of nodes and arcs in the graph. The number of optimization variables determines the size of the search space and affects how quickly MIP can converge to a solution, if any. However, when CF is introduced, the optimization problem and graph grow exponentially with the vehicle set cardinality. This is problematic because the search space becomes exponentially larger, making convergence to at least sub-optimal solutions more challenging. Limiting the maximum size of the coalition would result in smaller optimization variables (i.e. 2^m would be replaced by 2^{k_s} where $k_s << m$) but could still be challenging. Next, we propose QGA as a solver to address this challenge.

Table 4. Graph Complexity for 3-index MIP

	Optimization variables	Nodes	Arcs
Without CF	$(2n)^2 m$	$2n$	$(2n)^2$
With CF	$(2n)^2 2^m$	$2n$	$(2n)^2$

3.3. QGA Solver

Due to the larger search space of PDP-CF, we propose a QGA solver to efficiently traverse this exponentially large search space and possibly converge to a better solution than MIP. QGA adopts multiple mathematical constructs from quantum mechanics, including complex probabilities, state superposition and state collapse, to efficiently explore a search space and converge to a solution. In QGA, the 3-index MIP optimization variables, encoded in the chromosome, are viewed as existing in a linear superposition of their basis states, i.e. a binary randomly variable is simultaneously in a state of 0 and 1 until it is "observed". In this context, "observed" implies that the fitness function is computed; a random collapse is performed before the fitness of the chromosome can be computed. This collapse is influenced by the probability amplitudes of the state. Specifically, a random number, $r \in [0,1]$, is generated and compared to α^2. If it is larger, then the corresponding optimization variable is assigned a value of 1; otherwise, it is assigned a value of 0, as shown in (15). QGA's workflow is summarized in Table 5. Table 6 summarizes the rotation gate's adjustment strategy. x_i denotes the i^{th} bit in the chromosome and b_i denotes the i^{th} bit in the chromosome with the best fitness. $\delta\theta_i$ denotes the adjustment angle step and $s(\alpha_i, \beta_i)$ denotes the rotating angle direction. $f(x)$ and $f(b)$ denote the fitness of a chromosome and that of the best chromosome, respectively.

$$x_{ijk} = \begin{cases} 1 & \text{if } r > \alpha_{ijk}^2 \\ 0 & \text{if } r < \alpha_{ijk}^2 \end{cases} \tag{15}$$

$$\begin{bmatrix} \alpha_{i+1} \\ \beta_{i+1} \end{bmatrix} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) \\ sin(\theta_i) & cos(\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \tag{16}$$

Table 5. QGA Workflow

1. Initialize the population.
2. Collapse the chromosome to one of the basis states.
3. Compute fitness of the collapsed chromosomes.
4. While we have not converged:
    a. Select the rotation angle using Table 6.
    b. Apply the rotation gate to the chromosomes in the population, shown in (16).
    c. Apply the mutation operator.
    d. Apply the crossover operator.
    e. Collapse the chromosome to one of the basis states.
    f. Compute the fitness of the collapsed chromosomes.

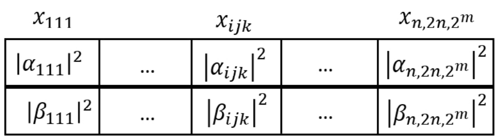
3.3.1. Encoding

Instead of encoding the binary optimization variables x_{ijk}, as in GA, QGA encodes the probability amplitudes of the optimization variable's states, as shown in Figure 3. We can view the binary variables as having one of two states,

Table 6. Quantum rotation gate angle adjustment strategy [28]

				$s(\alpha_i, \beta_i)$			
x_i	b_i	$f(x) > f(b)$	$\delta\theta_i$	$\alpha_i\beta_i > 0$	$\alpha_i\beta_i < 0$	$\alpha_i = 0$	$\beta_i = 0$
0	0	True/False	0	0	0	0	0
0	1	False	ω	+1	-1	0	±1
0	1	True	ω	-1	+1	±1	0
1	0	False	ω	-1	+1	±1	0
1	0	True	ω	+1	-1	0	±1
1	1	True/False	0	0	0	0	0

0 or 1, with complex probability α and β, respectively, such that $\alpha^2 + \beta^2 = 1$. We view the genes as a 2-dimensional vector with continuous real values representing the magnitude of the complex probabilities. The number of genes in the chromosome is equal to the number of optimization variables.



x_{111}		x_{ijk}		$x_{n,2n,2^m}$
$\lvert\alpha_{111}\rvert^2$	...	$\lvert\alpha_{ijk}\rvert^2$	...	$\lvert\alpha_{n,2n,2^m}\rvert^2$
$\lvert\beta_{111}\rvert^2$	...	$\lvert\beta_{ijk}\rvert^2$	...	$\lvert\beta_{n,2n,2^m}\rvert^2$

Fig. 3. QGA Chromosome encoding based on the 3-index formulation

3.3.2. Initialization, Mutation and Crossover Operations

The probability amplitudes are initialized to $\frac{1}{\sqrt{2}}$ for all chromosomes in the population, i.e. the probability of collapsing to state 1 or 0 is equal to 0.5. If we have some prior knowledge of which states would be better than others for each variable, we can initialize α and β accordingly. However, in this work, we assume no such prior knowledge.

The mutation operation must not violate the $\alpha^2 + \beta^2 = 1$ condition. Therefore, the mutation operator randomly modifies α only. Then, β is computed to satisfy the equality. The crossover operation is similar to traditional GA where a random cutoff point is selected, then genes from two parent chromosomes are swapped.

3.3.3. Fitness Function and Constraints

The fitness function, adopted in this formulation to evaluate the quality of a solution, is based on the objective function of the PDP-CF formulation. Specifically, the fitness of a chromosome is inversely proportional to the cost function which is the total distance traveled to deliver packages. This not only depends on the locations of the sources and destinations but also on the number of agents in a coalition. Furthermore, to incorporate environmental characteristics, we modified the fitness function to include a cost for traveling in difficult conditions. For example, an aerial vehicle would have a harder time traveling in windy conditions than a ground vehicle; a biped is prawn to failures on rocky terrains compared to six-legged vehicles. While some of these additional costs can be reflected in an energy cost function (as in the former), the increased probability of hardware failures (as in the latter) are not reflected. Therefore, we adopt a heuristic function, $f(x) = (1 + c(x)) \times d(x)$, that quantifies the difficulty (denoted by $c(x)$) a certain agent performing a task in a given set of conditions. We adopt a categorical function with 5 levels of difficulty: “very easy”, “easy”, “moderate”, “difficult”, and “very difficult”. Due to the difference in units between $c(x)$ and the distance cost function ($d(x)$), simply adding both cost functions is not suitable. Instead, we convert the 5 levels of difficulty into 5 values between 0 and 1 and multiply the distance cost function by the difficulty factor, i.e. we assume the distance is much larger due to the additional difficulty.

Since the PDP-CF formulation is a constrained optimization problem, part of the search space is in the infeasible region. Therefore, we need to insure that the chromosomes in the population are within the feasible region. To do so, we couple (or entangle in quantum mechanical terminology) the genes that would violate any of the constraints. For example, if x_{ijk} and $x_{ijk'}$ cannot be 1 simultaneously because coalitions k and k' would violate the coalition overlap constraint, then these two genes are coupled and the collapse of one would affect the value of the other. This coupling is performed on all genes that may cause the violation of any of the constraints, not just the coalition overlap constraints.

4. Experimental Results

4.1. Experimental Setup

The experiments were run on an Intel Core i7 processor with 8 GB of RAM and a Windows 10 operating system. The code was written in Matlab R2016b. In all experiments, the evolutionary algorithms were allowed to run for a maximum of 100 iterations, unless otherwise specified. Also, Manhattan distances were computed in the cost function. To test the proposed solvers, we randomly generated PDP scenarios by specifying the number of vehicles and packages, in addition to their respective capacity and demand ranges. We also specified the size of the grid that includes the pickup and delivery locations. We assumed all vehicles are housed in a depot with a fixed location (at the origin of the 2D grid). We also assumed that time windows are not imposed. PDP scenarios of various sizes were created and summarized in Table 7.

Table 7. Instance Description

Instance	Number of Requests	Demand Range	Number of Vehicles	Capacity Range
ID1	5	{1,2}	5	{1}
ID2	3	{1,2,3,4,5}	5	{1,2,3}
ID3	10	{1,2,3,4,5}	10	{1,2,3,4,5}
ID4	10	{1,2,3,4,5}	10	{1,2,3,4,5}
ID5	10	{1,2,3,4,5}	10	{1,2,3,4,5}

4.2. Performance Analysis

Table 8 summarizes the performance of QGA with and without CF. We notice that as the problem size increases, the computational time of QGA with CF increases exponentially while that without CF increases linearly. For small problem sizes, it is clear that QGA converges to less costly schedules. For larger problems, QGA struggles to converge to a solution (Inf implies an infinite cost function, i.e. solution is not in feasible region); this is expected given the exponentially larger search space. Therefore, it is important to develop a more computationally efficient search algorithm for PDP-CF.

Table 8. QGA Performance

	With CF		Without CF	
Instance	Total Distance	Running Time (sec)	Total Distance	Running Time (sec)
ID1	768	0.5142	1047	0.2207
ID2	466	0.2930	452	0.1907
ID3	2487	71.3716	2401	0.6433
ID4	Inf	69.0503	67329	0.6145
ID5	Inf	68.3353	2220	0.5559

5. Conclusion

In this work, we presented PDP-CF, a formulation that allows the formation of coalitions when delivering packages to achieve more efficient delivery schedules. A 3-index MIP formulation that searches the space of the vehicles' power set allows generated solutions to consider coalitions of cooperating vehicles. However, this formulation leads to an exponentially increasing search space. Hence, we proposed a QGA solver to leverage QGA's speed and efficient search space exploration, a byproduct of the chromosomes' linearly superposed gene states. Simulations on multiple PDP scenarios exhibited the effectiveness of the proposed encoding. Future work will consider more efficient MIP formulations by imposing additional assumptions such as restricting the number of agents per coalition. More complex PDP formulations with additional constraints will also be derived. We will also investigate the performance of the system when allowing coalition overlap while adopting an energy-based cost function. Incorporating non-vehicular agents should also be validated on realistic benchmarks.

Acknowledgements

This work was supported by the University Research Board at the American University of Beirut and the National Council of Scientific Research in Lebanon (CNRS-L).

References

[1] Al Chami, Z., Manier, H., Manier, M.A., Fitouri, C., 2017. A hybrid genetic algorithm to solve a multi-objective pickup and delivery problem. IFAC-PapersOnLine 50, 14656–14661.

[2] Bent, R., Van Hentenryck, P., 2006. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. Computers & Operations Research 33, 875–893.

[3] Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., Laporte, G., 2007. Static pickup and delivery problems: a classification scheme and survey. Top 15, 1–31.

[4] Berbeglia, G., Cordeau, J.F., Laporte, G., 2010. Dynamic pickup and delivery problems. European journal of operational research 202, 8–15.

[5] Coltin, B., 2014. Multi-agent pickup and delivery planning with transfers .

[6] Coltin, B., Veloso, M., 2014a. Online pickup and delivery planning with transfers for mobile robots, in: IEEE Int. Conf. Robotics and Automation (ICRA), IEEE. pp. 5786–5791.

[7] Coltin, B., Veloso, M., 2014b. Optimizing for transfers in a multi-vehicle collection and delivery problem, in: Distributed Autonomous Robotic Systems. Springer, pp. 91–103.

[8] Coltin, B., Veloso, M.M., 2014c. Scheduling for transfers in pickup and delivery problems with very large neighborhood search., in: AAAI, pp. 2250–2256.

[9] Dakroub, O., Boukhater, C.M., Lahoud, F., Awad, M., Artail, H., 2013. An intelligent carpooling app for a green social solution to traffic and parking congestions, in: Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on, IEEE. pp. 2401–2408.

[10] Furtado, M.G.S., Munari, P., Morabito, R., 2017. Pickup and delivery problem with time windows: a new compact two-index formulation. Operations Research Letters 45, 334–341.

[11] Ganesh, K., Narendran, T., 2008. Taste: a two-phase heuristic to solve a routing problem with simultaneous delivery and pick-up. The Int. J. Advanced Manufacturing Technology 37, 1221–1231.

[12] Garey, M.R., Johnson, D.S., 1979. Computers and intractability: a guide to NP-completeness. WH Freeman and Company, San Francisco.

[13] Jia, Z., Yu, J., Ai, X., Xu, X., Yang, D., 2018. Cooperative multiple task assignment problem with stochastic velocities and time windows for heterogeneous unmanned aerial vehicles using a genetic algorithm. Aerospace Science and Technology 76, 112–125.

[14] Kachitvichyanukul, V., et al., 2009. A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. Computers & Operations Research 36, 1693–1702.

[15] Lu, Q., Dessouky, M., 2004. An exact algorithm for the multiple vehicle pickup and delivery problem. Transportation Science 38, 503–514.

[16] Pankratz, G., 2005. A grouping genetic algorithm for the pickup and delivery problem with time windows. Or Spectrum 27, 21–41.

[17] Parragh, S.N., Doerner, K.F., Hartl, R.F., 2008a. A survey on pickup and delivery models part ii: Transportation between pickup and delivery locations. Journal für Betriebswirtschaft 58, 81–117.

[18] Parragh, S.N., Doerner, K.F., Hartl, R.F., 2008b. A survey on pickup and delivery problems part i: transportation between customers and depot. Journal für Betriebswirtschaft 58, 21–51.

[19] Reuters, 2016. How amazon is making package delivery even cheaper. URL: `http://fortune.com/2016/02/18/amazon-flex-deliveries/`.

[20] Ropke, S., Cordeau, J.F., 2009. Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science 43, 267–286.

[21] Savelsbergh, M.W., Sol, M., 1995. The general pickup and delivery problem. Transportation science 29, 17–29.

[22] Statista, 2015. Statistics and facts about global e-commerce. URL: `https://www.statista.com/topics/871/online-shopping/`.

[23] TriviaHive, 2016. On average, how many packages does amazon ship per day? URL: `http://triviahive.com/d/on-average-how-many-packages-does-amazon-ship-per-day-3D2A088A712354BE`.

[24] Ursani, Z., Essam, D., Cornforth, D., Stocker, R., 2011. Localized genetic algorithm for vehicle routing problem with time windows. Applied Soft Computing 11, 5375–5390.

[25] Wang, H.F., Chen, Y.Y., 2012. A genetic algorithm for the simultaneous delivery and pickup problems with time window. Computers & Industrial Engineering 62, 84–95.

[26] Xiao-feng, H., Liang, M., 2013. Quantum-inspired ant colony algorithm for vehicle routing problem with time windows. Systems Engineering-Theory & Practice 5, 1255–1261.

[27] Zhang, J.L., Zhao, Y.W., Peng, D.J., Wang, W.L., 2008. A hybrid quantum-inspired evolutionary algorithm for capacitated vehicle routing problem, in: International Conference on Intelligent Computing, Springer. pp. 31–38.

[28] Zhang, X., Jiang, D., Han, T., Wang, N., 2016. Feature dimension reduction method of rolling bearing based on quantum genetic algorithm, in: Prognostics and System Health Management Conference (PHM-Chengdu), 2016, IEEE. pp. 1–5.