A genetic column generation algorithm for sustainable spare part delivery: application to the Sydney DropPoint network

Michelle Dunbar, Simon Belieres, Nagesh Shukla, Mehrdad Amirghasemi, Pascal Perez, Nishikant Mishra

This is a post-peer-review, pre-copyedit version of an article published in Annals of Operations Research. The final authenticated version is available online at: https://doi.org/10.1007/s10479-018-2911-2

A Genetic Column Generation Algorithm for Sustainable Spare Part Delivery: Application to the Sydney DropPoint Network

Michelle Dunbar[a], Simon Belieres[b], Nagesh Shukla[c,*], Mehrdad Amirghasemi[d], Pascal Perez[d], Nishikant Mishra[e]

[a] *ACRF Image-X Institute, Sydney Medical School, University of Sydney, Sydney 2006, Australia*

[b] *l'Institut National des Sciences Appliques de Toulouse (INSA), 31077, Toulouse Cedex 4, France*

[c] *School of Systems, Management and Leadership, University of Technology Sydney, Sydney 2007, Australia*

[d] *SMART Infrastructure Facility, University of Wollongong, Wollongong 2522, Australia*

[e] *Hull University Business School, University of Hull, United Kingdom*

Abstract

Modern-day logistics companies require increasingly shorter lead-times in order to cater for the increasing popularity of on-demand services. There is consequently an urgent need for fast scheduling algorithms to provide high quality, real-time implementable solutions. In this work we model a spare part delivery problem for an on-demand logistics company, as a variant of vehicle routing problem. For large delivery networks, the optimisation solution technique of column generation has been employed successfully in a variety of vehicle routing settings and is often used in combination with exact methods for solving problems with a large number of variables. Challenges may arise when the pricing subproblem is difficult to solve in a realistic period due to complex constraints or a large number of variables. The problem may become intractable when the network structure varies daily or is known with less certainty over longer period. In such instances, a high quality heuristic solution may be more preferable than an exact solution with excessive run time. We propose an improved version of column generation approach integrating an efficient genetic algorithm to obtain fast and high-quality solutions for a sustainable spare parts delivery problem. More specifically, we propose to retain the traditional column generation iterative framework, with master problem solved exactly, but with pricing subproblem solved using a metaheuristic. Computational results on a real dataset indicate that this approach yields improved solutions compared to the current best-case business-as-usual costs. It also substantially decreases the computational time; allowing for high-quality, tractable solutions to be obtained in few minutes. We propose to strike a balance between the practical and efficient solution aspects of metaheuristic algorithms, and the exact decomposition and iterative aspect of the column generation solution technique.

Keywords: Spare parts delivery, Logistics, Optimisation, Metaheuristics

*Corresponding Author: Email: nagesh.shukla@uts.edu.au, Tel: +61 29514 2734

1. Introduction

As logistics companies create and adapt their services to cater to the ever-expanding, on-demand nature of modern business, a key component of their business model hinges on the optimal use of their resources for transporting goods from warehouse to customer. In addition, current business trends indicate the increased desire for creating environmentally sustainable services, to decrease carbon footprint and impact on the environment. Traditionally, logistics companies have evolved to service a relatively static set of customers with known demand at regular intervals; for which lead-times were sufficient large to allow for long-term planning and inventory. With the advent of just-in-time manufacturing and on-demand delivery services, lead-times have become increasingly short, and many modern logistics companies now service a customer base which may vary from day-to-day and whose location and demand may also be uncertain until a few hours prior to delivery.

For this reason, many real-world logistics companies, including those providing spare parts delivery services, are now seeking to develop high-quality vehicle routing solutions within shorter time intervals in order to reduce economic and environmental impacts, and remain competitive. Thus the development of efficient algorithms for computing high-quality vehicle routing solutions are becoming increasingly important. Some of the many operational costs that may be of importance to minimize are quantities such as total distance travelled, cost of the specific routing with respect to time-of-day traffic and travel-time considerations, fuel consumption and environmental footprint.

The traditional vehicle routing problem (VRP) formulation seeks to minimize the total distance or time cost using all vehicles, subject to the constraint that each customer is visited exactly once. These vehicles are initially located at a central depot, from which the products to deliver are collected and distributed to their respective customer, with each customer visited exactly once. The classical VRP was generalized by Solomon (1983) and Solomon (1987) to include customer preferences for delivery time via individual time windows that the vehicles must satisfy and is known as the Vehicle Routing Problem with Time Windows (VRPTW). Shukla et al. (2008) solved a theater distribution problem which represented a class of vehicle routing and scheduling problem. For a detailed survey of the literature surrounding the VRPTW problem and its variants, the reader is referred to Desrosiers et al. (1995) and Desaulniers et al. (2002). The VRP and VRPTW have many varied applications and extensions, involving both pick-up and delivery problem with time windows (PDPTW). For example, the taxi dispatch modelling of Wang et al. (2009) and the dial-a-ride Problem (DARP), which has traditionally been applied in the context of door-to-door transportation and services for the elderly or disabled, see Borndörfer et al. (1999) and the the dial-a-flight problem by Espinoza et al. (2008). Parragh (2011) extends this further to include a heterogeneous patient allocation model for ambulance configurations for the Austrian Red Cross. In case of supply chain configurations, Shukla and Kiridena (2016) have introduced a novel

approach for managing supply chain configurations in presence of environmental issues. In disaster response models of supply chains, various applications have been investigated earlier involving location planning (Kaur and Singh (2016), Elluru et al. (2017)). Similarly, a dynamic and sustainable procurement model for planning was also introduced recently in Kaur and Singh (2017).

Such real-world on demand services are often further complicated by problem specific constraints or the presence of binary variables which may result in the problem being excessively difficult or indeed intractable to solve in a realistic time frame for it to be of practical use for the operator. Consequently, heuristics have traditionally been employed in the solution of VRPTW and its variants, such as the insertion heuristic by Toth and Vigo (1997). Cordeau and Laporte (2007) propose a tabu search heuristic for the static DARP with route duration constraints that evaluates the capacity of delaying the departure time from the depot to help improve solution feasibility. Pacquette et al. (2013) formulate a metaheuristic, integrating a reference point method for multicriteria optimisation within a tabu search algorithm.

Metaheuristics and evolutionary algorithms may be viewed as intelligent agents traversing a very large space of solutions, through biased random sampling. Thus, imposing additional constraints may reduce the size of this search space and, consequently reduce the search complexity. However, compared to exact approaches, these methods do not guarantee optimality, and can only achieve the goal of producing high quality, near-optimal solutions in a stipulated time frame.

The key contributions of this work are threefold:- Firstly, we propose a novel framework for solving a large-scale, on-demand spare-parts delivery problem; a hybrid solution approach utilizing the benefits of both column generation and metaheuristics. Secondly, our metaheuristic is easily customizable, and capable of handling complex constraints and adapting for a network that changes on a daily-basis. Thirdly, our hybrid approach enables the fast and efficient solution of a real-world network in under a few seconds, within 0.1%-9.6% of optimality (and an average 2.9%).

The rest of this paper is organised as follows. Section 2 presents the basic formulation for the spare parts delivery problem based on VRP and Section 3 presents a brief survey of exact and metaheuristic strategies for the spare parts delivery problem. Section 4 presents the proposed hybrid methodology and also provides some computational results. Finally, concluding remarks and some directions for future research is outlined in Section 5.

1.1. The Drop-Point Problem

Companies that utilise field technicians have traditionally followed a classical supply chain model for the delivery of goods, that in an on-demand setting becomes inflexible and impracticable as service areas become geographically large. The logistics company *DropPoint* with whom we collaborate provide a disruptive solution in which technicians reduce the amount of travel to and from sparsely located warehouses to receive and return job-specific tools, and thus not only reduce lost productivity and excess travel time, but also reduce the need for storing excess inventory. A schematic of the DropPoint network is displayed in Figure 1 below:

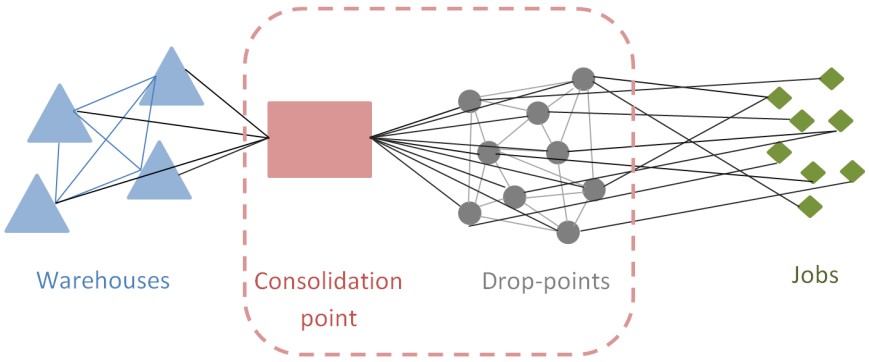

Figure 1: A Schematic of the Drop-Point Network.

The DropPoint supply chain consists of four key components (see Figure 1). Initially the required job-specific tools are located within dedicated warehouses that are distributed throughout the metropolitan area. These tools are then transferred to a *consolidation point*, from which they are distributed overnight to a set of locations known as *drop-points* (eg. 24hr Service stations, convenience stores), located close to the job requested by the technicians. The technicians are then able to pick-up the necessary tools from the drop-point(s) rather than from the warehouses themselves. This dramatically reduces unnecessary travel by the technicians, often across the CBD, during peak hours.

The specific problem of interest is that highlighted in Figure 1 - the process of distribution from the consolidation point to the individual drop-points. The objective of the company is to both reduce the travel time required to deliver the goods from the consolidation point to the drop-points within specified time windows, and also reduce its carbon footprint. Given the complexity of this delivery to the drop-points, the delivery of goods from the consolidation point to each of the drop-points was identified by the company as the component most able to benefit from optimisation. We thus focus the remainder of this work on this specific application.

2. Problem Formulation

The spare-parts delivery problem outlined above is formulated as a variant of VRPTW and there exists a number of different solution approaches for solving the VRPTW and its variants. These include flow-based and path-based integer programming models, solved using exact algorithms and greedy heuristics. Since this problem is naturally characterized as a path-based model, we pursue this approach in a column generation setting, utilizing a heuristic to provide a fast solution for the specific network defined on the day of operations.

As the scale of such a network can grow rapidly in large metropolitan cities, we seek to solve this problem efficiently by proposing to strike a balance between the practical and effi-

cient solution aspects of metaheuristic algorithms, and the exact decomposition and iterative aspect of the column generation solution technique. More specifically, we propose to retain the traditional column generation iterative framework, with master problem solved exactly, but with pricing subproblem solved using a metaheuristic. Moreover, our metaheuristic will be tailored so as to keep track of both cost and reduced-cost, ensuring that we generate only negative reduced cost columns to add to the master problem at each iteration.

The formulation that we present in this section is based on the master problem/subproblem formulation used in column generation. Our proposed problem is stated as follows. Given a set of m nodes (locations known as drop-points), each node is required to be visited exactly once by one vehicle in a minimal cost way subject to the following additional constraints. Each node i may only be visited within a specified time window $[a_i, b_i]$, and a certain demand D_i at each location must be fulfilled in a single delivery. For the particular real-world logistic company, many of these time windows overlap meaning that there is no natural way to impose an ordering on the nodes. Thus, we assume in this work that the nodes are unordered, and that the labelling of nodes is purely for identification purposes.

The path taken through the network by each vehicle j, may be stored as a column vector $\mathbf{a}_j$, for which each element a_{ij} is defined as follows:

$$a_{ij} = \begin{cases} 1 & \text{if node } i \text{ is visited in path } j \\ 0 & \text{otherwise} \end{cases},$$

It is assumed that all vehicles start and end their route at a central depot, hereafter referred to as a *consolidation point*. Each vehicle j has an associated route cost c_j equal to the travel time/distance accumulated along each route which is a combination of operational costs and environmental impact of the delivery. We also ensure that each node is visited exactly once by exactly one vehicle using a set-partitioning constraint. This gives rise to the following optimisation problem.

This gives rise to the following master problem:

$$\text{Minimize:} \quad \sum_{j=1}^{N} c_j x_j \tag{1}$$

$$\text{Subject to:} \quad \sum_{j=1}^{N} a_{ij} x_j = 1 \quad \forall\, i = 1, \ldots, m \tag{2}$$

$$x_j \in \{0, 1\} \tag{3}$$

We now present the pricing subproblem for generating vehicle routes to add to the master problem. As will be discussed in Section 4, a genetic algorithm, instead of an ILP solver, is implemented to solve the subproblem and the presentation of the ILP formulation here is just for the purpose of mathematical clarity. We define the cost of travelling between node i and j as c_{ij}, and the decision variable X_{ij} as follows:

$$X_{ij} = \begin{cases} 1, & \text{if node } i \text{ and } j \text{ are connected in a path} \\ 0, & \text{otherwise} \end{cases}$$

To construct a path, we define two dummy nodes, a source s and a sink t which both correspond to the consolidation point; the beginning and end of the cycle. We assume homogeneity of all vehicles and thus an equal capacity of Q units per vehicle. In order to keep track of the ordering of the nodes in each path, we introduce integer variables u_i which denote the position of node i in the path, and we introduce continuous variables S_i which denote the time at which the vehicle arrives at node i. We denote the travel time between node i and node j by the constant t_{ij}.

To find the best possible path to add to the master problem, we are required to determine the path with most negative reduced cost. Thus in each iteration, we make use of the dual values μ_i corresponding to constraint i in the the master problem in order to find the path with minimum reduced cost as our objective in the pricing subproblem.

$$\text{Minimize:} \quad \sum_{i=1}^{m}\sum_{j=1}^{m}[(c_{ij} - \mu_j).X_{ij}] \tag{4}$$

$$\text{Subject to:} \tag{5}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{m} D_j X_{ij} \leq Q \tag{6}$$

$$\sum_{j=1}^{m} X_{sj} = 1, \quad \forall j \in \{1, \ldots, m\} \tag{7}$$

$$\sum_{i=1}^{m} X_{it} = 1, \quad \forall j \in \{1, \ldots, m\} \tag{8}$$

$$\sum_{i=1}^{m} X_{ij} \leq 1, \quad \forall j \in \{1, \ldots, m\} \tag{9}$$

$$\sum_{i=1}^{m} X_{ij} - \sum_{k=1}^{m} X_{jk} = 0 \quad \forall\, j \in \{s, 1, \ldots, m, t\} \tag{10}$$

$$(m+1)X_{ij} - m \leq u_j - u_i \leq m - (m-1)X_{ij}, \quad \forall j \neq s \tag{11}$$

$$S_i - S_j \leq b_i - a_j - X_{ij}(b_i - a_j + t_{ij}), \quad \forall j \neq s \tag{12}$$

$$S_i - S_j \geq a_i - b_j - X_{ij}(a_i - b_j + t_{ij}), \quad \forall j \neq s \tag{13}$$

$$X_{ij} \in \{0, 1\}, u_i \in \{0, 1, \ldots, m\}, S_i \in \mathbb{R} \tag{14}$$

Constraint (6) ensures that the capacity of each vehicle does not exceed Q units. Constraint (7) and (8) ensure that each vehicle visits the consolidation point only to depart and return exactly once. Constraint (9) ensures that each feasible connection between nodes is used at most once, and constraint (10) ensures flow balance. Constraint (11) ensures that

there is only one cycle which includes the consolidation point, and constraints (12) and (13) ensure that each drop-point is visited within its specified time window.

It is worth noting that the VRP has strong similarities to the Job-shop scheduling problem (JSP). In the JSP, as a problem with applications in advanced manufacturing systems, a number of jobs have to be processed on a number of machines, with each job having specific processing time on each machine. The objective is to find an order of jobs on each machine, so that the completion time of the last job on the last machine, also known as *makespan*, is minimized.

In a relaxed way, we can view the *jobs* in JSP as the equivalent of *vehicles* in the VRP, and the *processing times* in the JSP as the equivalent of the *pair-wise distance* between locations in the VRP. A reformulation of the JSP, as a VRP has been presented in (Beck et al., 2002). Both the VRP and the JSP can be considered as a generalization of the Travelling Salesman Problem (TSP). The success of applying evolutionary algorithms in general, and Genetic Algorithms in particular, to Job-shop and (permutation) Flowhsop Scheduling problems (PFSP), as demonstrated in (Amirghasemi and Zamani, 2015, 2017) was one of the key motivation for incorporatings the GA for the VRP in this study.

3. Related Work

VRP and its variations have been widely studied in the literature during the past decades. In this section, the approaches used to solve the VRP are classified into two categories of exact and metaheuristic methods. While the exact methods are typically branch and bound techniques which guarantee finding an optimal solution, the metaheuristics focus on finding high-quality solutions quickly. In each section we will outline the relevant studies in chronological order, and focus on the basic VRP with some minor references to VRPTW.

3.1. Exact Methods

The last few years have witnessed a growing body of literature focusing on the development of exact solution methods, and exact methods used in combination with efficient heuristics. Exact algorithms used to solve the VRPTW and its variants are primarily based on branch-and-cut and branch-and-price algorithms, used in combination with an efficient labelling algorithm to solve the resource constrained shortest path problem. Recent contributions include Ropke et al. (2007) who propose a branch and cut algorithm for the PDPTW based on a two-index MIP model and applied it solve the DARP effectively. Baldacci et al. (2011) also propose a pickup and delivery problem with time windows, and present a new exact algorithm for the based on a set-partitioning-like integer formulation, and a heuristic bounding procedure that finds a near-optimal dual solution of the LP-relaxation.

3.2. Metaheuristics

Metaheursitics can be categorized into three broad classes:- construction, point-based, and pool-based methods. Construction methods are randomized methods that generate a solution in an incremental fashion by deciding the value of each solution component one at a time. Ant Colony Optimisation (ACO) for instance, belongs to this category. Point-based methods however receive a single, complete solution as the input and iteratively improve the given solution until some stopping criterion is met. Local searches and their variations belong to this class. Finally, pool-based methods aim to improve the quality of a pool of solutions through combining high-quality solutions in the pool, in order to achieve higher quality solution. Evolutionary Algorithms (EAs) in general and Genetic Algorithms (GAs) in particular belong to this category.

Since the most relevant approaches to our approach are pool-based techniques, we mainly focus on Evolutionary Computation in general and Genetic Algorithms in particular. The interested reader can see (Shukla et al., 2013b), (Shukla et al., 2013a), (Bräysy and Gendreau, 2005) and (Potvin and Bengio, 1996) for a full survey on all metaheuristics addressing the VRP. In the following a brief review on evolutionary computation is presented first, next effective GAs tackling the VRP are reviewed.

The Evolutionary Computation paradigm encompasses a wide variety of techniques (De Jong, 2006). These techniques can be divided in two broad categories. The first group, Evolutionary Algorithms (EAs), are adaptive systems primarily inspired by the theory of biological evolution. The second group, which are often inspired by self-organized, decentralized swarming behaviors of natural species, are Swarm Intelligence (SI) techniques such as Ant Colony Optimization (Dorigo and Gambardella, 1997) and Particle Swarm Optimization (Clerc and Kennedy, 2002).

EAs can be viewed as agent-based models wherein each agent represents a candidate solution to the problem being solved. These agents *compete* and *cooperate* and the resulting emergent complexity is a population dominated by the highly *fit* agents. The competition is performed via the *competitive selection* and the cooperation is obtained through genetic *recombination operators*. The *fitness* is basically defined as the agent's relative "goodness" in the population. Here the mechanics of EAs is explained in detail, based on the unified framework presented in (De Jong, 2006).

In the terminology used in EA community, these "solution agents" are referred to as *individuals*, and are represented by a real or binary-valued vector. This vector is also known as a *chromosome*. The essential procedures which characterize a simple EA include (i) an encoding/decoding scheme that maps every solution (*phenotype*) to a chromosome (*genotype*), (ii) a *fitness function* that assigns a goodness to each individual, (iii) a *parent selection* strategy which determines which individuals are nominated as parents to produce offspring, (iv) a *survival selection* strategy in which a rule is defined for deciding which individuals will be survived to the next generation, and (v) *reproduction operators*, which specify how new offspring will be produced from parents.

A generalized pseudocode for an EA with a parent population size m and offspring population size n is presented in Figure 2.

Procedure EvolutionaryAlgorithm
begin
// Initialization
Generate m random individuals as the initial population;
do
    //Parent Selection:
    select k individuals ($k \leq m$) as the set of parents, P;
    // Generating Offspring:
    Given the set P, produce a set of n offspring, O, using reproductive operators;
    //Survival Selection:
    From the set $T := P \bigcup O$ select m individual to survive to the next generation;
    Replace the current population with the m newly selected individuals;
while (some Stopping Criterion is met);
return the best solution in the population
end.

Figure 2: A general framework for simple EAs

Implementing the general framework presented in Figure 2 will characterize a simple evolutionary algorithm. In Evolutionary Programming (EP), the parent and offspring population size are equal ($m = n$), each parent produces one offspring and survival selection method is simply selecting the m individuals with the highest fitness (De Jong, 2006).

In Evolution Strategies (ES), another canonical EA, the population size is generally chosen to be smaller than the offspring population size. For example, when $m = 1$ and $n = 10$, one parent produces 10 offspring, and from the combined pool of 11 individuals only the one with the highest fitness survives. The only reproduction operator in ES is an *asexual* reproduction operator, namely a Gaussian *mutation* operator that perturbs the solution based on the Gaussian (Normal) distribution (De Jong, 2006).

Genetic Algorithms (GAs), developed by Holland (1975) are the most well-known in the EA literature. In standard GA, the binary representation is used and the parent selection is proportionally biased towards the individuals with higher relative fitness. Each two selected parent produce two children using a *sexual* (two-parent) reproduction operator called *crossover*. In simple one-point crossover operator, both two individuals chromosomes are divided at one point and the sub-chromosomes are exchanged to produce two offspring. The crossover operator is illustrated in Figure 3.

parent 1:	**1 1 0**	**1 0 1 0 1**
parent 2:	0 1 1	0 1 1 1 0

offspring 1:	**1 1 0**	0 1 1 1 0
offspring 2:	0 1 1	**1 0 1 0 1**

Figure 3: One-point crossover applied to two binary chromosomes

In general, the standard GA starts with a population of m individuals; m new offspring are produced using crossover, and then, a *bit-flip* mutation operator, which perturbs each individual probabilistically (one bit per chromosome on average), is applied. Finally, the new m individuals replace the old individuals, forming the next generation.

In all effective evolutionary algorithms, the following technical considerations are essential:- First, the encoding/decoding procedures should allow for fast evaluation of individuals. Second, reproduction operators should be selected in such a way that offspring inherit good characteristics from their parents. Third, there must be a proper balance between *exploration* and *exploitation*; as outlined by Eiben and Schippers (1998), both selection schemes and choice of reproduction operators do influence the explorative and exploitative power of the search.

When employing the GA for modeling a combinatorial optimization problem, such as JSP, or VRP, a *permutation* representation could be used instead of the binary encoding. For these permutations, special crossover and mutation operators are required. The crossover operators are designed in a way that parents can pass their common characteristics, such as adjacency, precedence, and position, to their offspring. Table 1 presents some of the effective permutation crossover operators currently used in the literature. For permutations, a path

Table 1: The major permutation crossover operators

Permutation Crossover Operator	Paper
OX, 1X	(Davis, 1985)
OX2, PBX	(Syswerda, 1991)
PMX	(Goldberg and Lingle, 1985)
TBX1, TBX2, HTBX, UX2	(Poon and Carter, 1995)
ERX	(Whitley et al., 1989)
EERX	(Starkweather et al., 1991)
LOX	(Falkenauer and Bouffouix, 1991)
PPX, GOX, GPMX	(Bierwirth et al., 1996)
LCS	(Iyer and Saxena, 2004)
kX, TDX	(Djerid et al., 1996)
IX, UX	(Fox and McMahon, 1991)

relinking operator is characterized by the *move operator* which is defined on the space of permutations. For example, a *swap move* swaps two elements by changing their orders,π_i,π_j, and a *shift move*, shifts an element to a new position by changing the order of elements whose positions change.

Applying path relinking with different move operators has the potential to be an effective alternative to classic permutation crossover operators; albeit at the cost of longer run-times. Compared to permutation crossover operators, path relinking has more explorative and exploitative power. This is due to the fact that it considers all solutions in-between two

solutions and selects the best solution. The idea of producing an optimized offspring from two parents is developed by Aggarwal et al. (1997) for solving Maximum Independent Set problem with GAs. Path relinking has been successfully applied to the QAP (Ahuja et al., 2000) and the PFSP (Reeves and Yamada, 1998).

It is worth noting that most studies typically propose a combination of pool-based and point-based techniques as effective hybrids. One of these well-known hybrids is the Memetic Algorithms (MAs) (Moscato, 1989) wherein a pool-based and point-based methods are simply integrated. In MAs, a pool-based method, such as local search, is applied before and/or after applying crossover and mutation operators. Examples of effective application of memetic strategies to JSP and PFSP are (Gao et al., 2011) and (Amirghasemi and Zamani, 2017).

3.3. Genetic Algorithms for Solving the VRP

One of the pool-based strategies for the vehicle routing problem is the work of (Rochat and Taillard, 1995) which can be used to improve any local search techniques for both the VRP and VRPTW. In the proposed method, initially a number of initial solutions are generated using a local search. Next, as each generated solution consists of a set of paths, from the combined set of paths belonging to all solutions, high-quality paths are selected, in a randomized fashion, for the purpose of forming a new solution. In effect, in this randomized scheme paths belonging to higher quality solutions have a higher chance of being included in the new solution. It is worth noting that since some locations may not be visited in the newly generated solution the solution can be infeasible. To overcome this issue, the remained locations are treated as an independent VRP problem which are solved by a local search. Finally, the newly generated solution undergoes a local search and the paths belonging to that solution is added to the pool of paths. This procedure continues until a stopping criterion is met.

One of the key issues in designing any Genetic Algorithm is the encoding/decoding scheme (Baker and Ayechew, 2003). In the encoding proposed in this work, each DropPoint is assigned to a vehicle (path). For instance, for a problem with 10 locations, a solution can be represented as:

Locations:	1	2	3	4	5	6	7	8	9	10
Solution 1:	1	1	2	2	2	1	3	3	2	1

Thus this encoded solution **1122213321**, can be described as follows:- The 10 locations are serviced by three vehicles, and, vehicle 1 services locations 1,2,6, and 10. Similarly, vehicle 2 services locations 3,4,5, and 9. Several effective crossover and mutation operators can be defined on such encodings. Baker and Ayechew (2003) provide computational experiments on the benchmarks extracted from OR-Library (Beasley, 1990) and conclude that integration of tabu search and simulated annealing with their proposed approach gives the best result.

Our proposed procedure, named Genetic Column Generation (GCG), can be seen as a

hybrid of exact methods and the GA as a metaheuristics. The rationale that this integration is effective is twofold. First, specific problem domain knowledge is incorporated, assisting the GA through improving the quality of the individual solutions selected for reproduction operators. Second, incorporation of column generation leads to a higher exploitative power of the algorithm, while maintaining the explorative benefits offered by the GA, simultaneously. In the following subsection, some hybrid methods are briefly reviewed.

3.4. Hybrids

Beheshti and Hejazi (2015) proposed the hybridization of the column generation method and metaheuristics via *collaborative* and *integrative* schemes. A Quantum Inspired Evolutionary Algorithm (QIEA) was used to solve the subproblem and the Electromagnetism-like algorithms have been incorporated as an alternative to column generation. Compared to our proposed procedure, we these methods possess a large degree fine-tuning in the parameters, hence limiting their applicability in our on-demand problem setting. *SearchCol* was proposed by Alvelos et al. (2013) as a general framework for combining metaheuristics with column generation. SearchCol aims to exploit the knowledge of all the subproblems generated by the master problem in each iteration and using this knowledge to do a *perturbation* to an incumbent solution. To the best of our knowledge SearchCol has not been practically applied to the classic VRP with time-windows.

4. The Genetic Column Generation

Column Generation (CG) can be seen as a decomposition technique in which solutions are incrementally improved through the interactions between the *Restricted Master Problem* (RMP) and a *subproblem*. Initially a Restricted Master Problem (RMP) considers a subset of feasible routes is solved. The dual information from this RMP is then used to inform a subproblem, so as to produce a route with negative reduced cost, to add to the RMP so as to improve the overall solution. This process continues until no more routes with negative reduced cost can be proposed by the subproblem, and the problem terminates at the optimal solution. Figure 4 shows a simplified pseudocode of the CG process.

Our proposed Genetic Column Generation (GCG) follows the same master problem, subproblem iterative structure. However, as our subproblem has a large number of variables and requires significant computational time to solve for a large-scale network, we propose to solve the subproblem using a Genetic Algorithm. This also affords the company additional flexibility in adding in and modifying constraints on the day of operations.

This proposed Genetic Algorithm is to be VRPTW-specific and has the objective of determining the path with most negative reduced cost. For this purpose, the algorithm will be warm-started with an initial pool (population) of arbitrary paths and utilize problem-specific *selection mechanisms* and *reproductive operators* to generate beneficial paths in each iteration. This is achieved by using the *reduced-cost objective function* as its *fitness function*;

```
Procedure Column Generation
begin
// Initialization
 Initialize  data structures for the master problem and subproblem;
do
    Solve the master problem and compute the dual values;
    Update the objective function of the subproblem based on the computed dual values;
    Solve the subproblem and compute the objective function value as X;
    if X<0 add the respective column (path) to the master problem;
while (X<0);
Return the Master Problem objective function value as the best obtained cost;
end.
```

Figure 4: A general pseudocode of Column Generation procedure

meaning that not only is it able to find a high quality path quickly, but provides us with the multiple alternative options to choose from (with varying levels of solution quality) for a specified number of iterations.

Our proposed CGA, however, is not a canonical GA (De Jong, 2006) for two reasons. Firstly, instead of generating a new population of paths in each iteration, only the best generated path is returned. Secondly, in each iteration the Master Problem Solver is called to update the dual values. Figure 5 shows the pseudocode of the Genetic Algorithm implemented to solve the subproblem.

```
Procedure Genetic Subproblem Solver
begin
 Initialize  the  initial  population of paths based on the information received from the master problem;
for ( number of paths in the initial  population ) do
  begin
    Select two random paths and generate two offspring using the crossover operator;
    Mutate the generated path based on the given mutation probability;
    if the newly generated paths respect the constraints, add them to the current population;
  end
return the path having highest fitness(minimum reduced cost) as the output;
end.
```

Figure 5: The psuedocode of Genetic Subproblem solving procedure

This is incorporated in the Column Generation framework presented in Figure 4 and eliminates the need for calling an external Integer Linear Programming (ILP) solver. In the next subsection, the specifics of this incorporated Genetic Subproblem Solver are described.

4.1. The GA settings

In this subsection we outline our proposed encoding, construction of the initial population, crossover and mutation operators, and fitness function.

4.1.1. Encoding scheme

We model the paths as a binary string with of size n, the number of drop-points to visit. The i^{th} bit in the binary string takes the value of 1 if the path visits the i^{th} drop-point and 0 otherwise. This encoding ensures that several constraints, namely,(i) the flow conservation, (ii) the fact that each node is visited at most once, and (iii) avoiding short cycles, are satisfied. In addition, crossover and mutation operators can be easily applied to this binary encoding. However, the only drawback of this encoding method is that it does not specify the order in which the drop-points are visited. To rectify this issue, we will outline a randomized heuristic procedure for determining the visiting order in subsection 4.1.4 below.

4.1.2. The initial population

At each iteration, an initial population is required for the Genetic subproblem solver. Since the initial population of a GA strongly influences its effectiveness, an problem-specific approach needs to be adopted. In our implementation, we use the current collection of paths used to build the matrix of constraints in the Master Problem, as the initial population of the Genetic subproblem solver Algorithm. In effect, a collection of high-quality paths provided by domain or industry experts, could also be the inform the construction of the initial population.

4.1.3. Crossover and mutation

As the two reproduction operators, crossover and mutation have the key role of diversifying the population. For the crossover operator, a 2-point crossover is applied on each pair of uniformly random selected paths. Two integers (c_1, c_2) are generated for each pair, which will mark the binary string in two locations where the crossing happens. For instance, consider a simple pair of paths, with 12 drop-points:

$$\begin{array}{lcccccccccccc} \text{parent 1:} & (& 0, & 1, & 0, & 1, & 0, & 1, & 0, & 1, & 1, & 1, & 0, & 0 &) \\ \text{parent 2:} & (& \mathbf{1}, & \mathbf{1}, & \mathbf{0}, & \mathbf{0}, & \mathbf{0}, & \mathbf{0}, & \mathbf{1}, & \mathbf{1}, & \mathbf{1}, & \mathbf{1}, & \mathbf{1}, & \mathbf{1} &) \end{array}$$

If for instance we have $c_1 = 2$, and $c_2 = 7$, this means that the position between the 2^{nd} and 3^{rd} bit, together with the one between the 7^{th} and the 8^{th} bit, will be the dividing spots:

$$\begin{array}{lcccc|ccccc|cccc} \text{parent 1:} & (& 0, & 1, & 0, & 1, & 0, & 1, & 1, & 1, & 0, & 0 &) \\ \text{parent 2:} & (& \mathbf{1}, & \mathbf{1}, & \mathbf{0}, & \mathbf{0}, & \mathbf{0}, & \mathbf{0}, & \mathbf{1}, & \mathbf{1}, & \mathbf{1}, & \mathbf{1}, & \mathbf{1} &) \end{array}$$

Finally the sub-vector between the crossover points are exchanged, producing two new offspring:

$$\begin{array}{lcccc|ccccc|cccc} \text{offspring 1:} & (& 0, & 1, & \mathbf{0}, & \mathbf{0}, & \mathbf{0}, & \mathbf{0}, & \mathbf{1}, & 1, & 1, & 1, & 0, & 0 &) \\ \text{offspring 2:} & (& \mathbf{1}, & \mathbf{1}, & 0, & 1, & 0, & 1, & 0, & \mathbf{1}, & \mathbf{1}, & \mathbf{1}, & \mathbf{1}, & \mathbf{1} &) \end{array}$$

The mutation operator is simply defined as a bit-flip operator: every bit of the offspring could change from 0 to 1 (or from 0 to 1) with the probability of $p = \frac{1}{2n}$.

4.1.4. Random algorithm to generate orders

In a population of paths, each offspring path indicates which drop-points are visited, and which are not. However, a cost (or fitness) has to be assigned to each path and depends on the order in which the drop-points are visited. The problem of finding the best ordering of each path is equivalent to solving the well-known Travelling Salesman Problem (TSP), known to be an $\mathcal{NP}$-hard problem. Due to the on-demand computational time constraints imposed by the company, we opt to replace a computational expensive exact-procedure with a heuristic nearest-neighbor scheme.

This simple nearest-neighbor algorithm is a 2-opt nearest neighbor algorithm, which stochastically generates an order for each path. It starts from the consolidation point and checks what are the two closest drop-points, in term of cost. Next, given a probability p, there is $p\%$ chance that the "closest" drop-point is selected, and $(1-p)\%$ chance for choosing the other one. The selected drop-point is added to the route, and we proceed in the same way considering its two closest drop-points. Figure 6 shows how this random generation of orders is performed for a small problem with 3 drop-points, with $p = 0.5$. This procedure is repeated until the entire drop-points are visited, leading to a complete path with a total cost c.

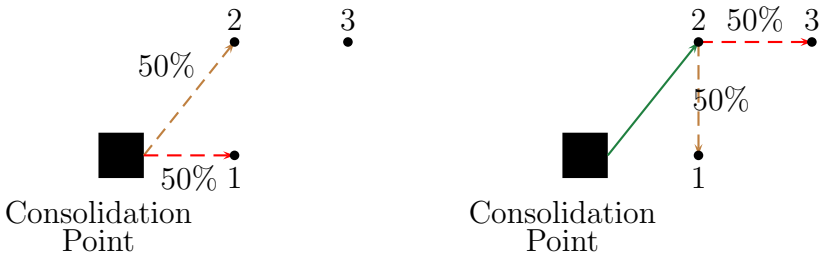

Figure 6: The stochastic generation of orders

4.1.5. The fitness function

The fitness function is calculated for each offspring, and allows us to quantify the value of each path. As we combine column generation and the Genetic Algorithm to optimise our algorithm, we take into account the Master Problem dual value for each constraint in this function. Thus, this value will inform us whether the solution to the subproblem will improve the objective of the Master Problem. If we consider an offspring $\mathbf{off}_i$ with a cost c_i, and a dual $\mathbf{d} = (d_1, \ldots, d_n)^T$ the fitness function is defined as:

$$f(\mathbf{off_i}, \mathbf{d}) = c_i - \mathbf{off_i}^T \mathbf{d}$$

This value is the reduced cost associated to the path. If this reduced cost is negative, the offspring is added to the Master Problem collection of paths. The lower this reduced cost, the greater the improvement.

5. Computational Experiments

Both Column Generation (CG) and Genetic Column Generation (GCG) are programmed in MATLAB and the IBM ILOG CPLEX solver API has been employed as the the Integer Linear Programming (ILP) solver. While the ILP solver is called for solving both the master problem and subproblem in CG, in GCG it is only used for solving the Master problem. The experiments have been run on Linux operating system on a single AMD Opteron CPU with 2.3 GHz speed.

We generated 40 random problem instances named inst01-inst40, with 20,30,40 and 50 drop-points. For each class of a specific size, 10 random problems were generated, in which each drop-point location is defined by an (x, y)-pair where x and y are uniformly random and $0 < x, y < 100$. Each problem instance was run once with the CG procedure and 10 times with the GCG procedure to obtain an average result. Final solution values (Cost) as well as the running times in minutes, are presented in Table 2. Due to the fact that GCG is an stochastic procedure, Best and Average Cost and Running times, in 10 runs, have also been reported.

As can be seen in Table 2, GCG universally outperforms CG in terms of the required running times. It may also be observed that in 12 out of 40 instances (30%), our GCG also achieves a smaller cost. For example inst28 CG returns a cost of 933.59 in over 164 minutes, but our GCG creates a solution with an even lower cost of 912.02 in under than a minute. These instances are indicated with bold font in Table 2. Moreover, the average running time for all instances in under one minute for the GCG procedure, whereas the classical CB varies from between 1.5s-19min. This fast solution time and ease of practical implementation was considered highly desirable by the company and was considered beneficial, outweighing any additional minutes prescribed by the GCG solution, as they were sufficiently close to the solution obtained by the classical CG in these remaining cases.

The proposed GCG algorithm was also implemented to solve a real case study data containing 76 drop-points and a consolidation point. This problem was initially solved using the exact formulation. However, the exact solution approach was not solvable in our 8 hour real-world time frame; requiring over 6 hours to achieve a solution to within 10% of optimality for a smaller instance of 26 nodes. Furthermore, the exact formulation did not make any significant progress on the real-world problem of 76 nodes within the 8 hour time limit. Thus we report the solution for the GCG algorithm only. The best result obtained after running GCG on the droppoint data is shown in Table 3. There were 16 trips from consolidation point (CP) to the drop-points (1 to 76). The average costs obtained for the routes is 1073 minutes and required only 7.6 minutes of computation time. This is of significant importance for the

Table 2: The comparison of CG and GCG procedures, with respect to cost and running time

		CG		GCG			
problem	**size**	**Cost**	**Time (m)**	**Best Cost**	**Best Time**	**AVG Cost**	**AVG Time**
inst01	20	604.97	1.98	617.36	0.30	640.70	0.32
inst02	20	472.23	2.28	472.55	0.30	486.70	0.33
inst03	20	**577.45**	2.08	**577.45**	0.28	604.80	0.33
inst04	20	**534.80**	2.10	**518.43**	0.29	535.07	0.32
inst05	20	**436.77**	1.84	**436.77**	0.29	448.08	0.34
inst06	20	531.36	2.32	532.29	0.29	557.17	0.31
inst07	20	**594.38**	1.00	**587.37**	0.30	626.11	0.33
inst08	20	**580.28**	1.74	**580.28**	0.29	626.43	0.32
inst09	20	565.98	1.60	568.47	0.31	581.59	0.33
inst10	20	**542.23**	1.57	**532.99**	0.29	560.43	0.33
inst11	30	722.68	11.58	730.61	0.42	746.35	0.44
inst12	30	655.29	18.24	693.14	0.41	723.62	0.43
inst13	30	685.31	14.85	694.99	0.38	744.87	0.44
inst14	30	**797.70**	20.71	**771.75**	0.43	796.25	0.46
inst15	30	**834.10**	8.68	**833.23**	0.42	853.78	0.44
inst16	30	718.44	55.96	741.79	0.40	762.47	0.44
inst17	30	726.73	15.20	757.40	0.42	776.79	0.45
inst18	30	**755.15**	28.62	**735.48**	0.42	768.35	0.44
inst19	30	**733.08**	33.76	**722.64**	0.40	749.73	0.44
inst20	30	**778.16**	6.88	**775.51**	0.41	815.82	0.43
inst21	40	971.81	126.43	1054.70	0.51	1101.06	0.53
inst22	40	921.87	136.22	941.84	0.52	986.70	0.53
inst23	40	1019.20	164.51	1078.10	0.50	1111.55	0.52
inst24	40	905.44	313.57	908.53	0.49	968.85	0.52
inst25	40	896.43	206.32	956.27	0.49	993.89	0.52
inst26	40	880.08	582.92	952.07	0.50	995.06	0.52
inst27	40	978.54	141.76	1024.60	0.51	1075.34	0.53
inst28	40	**933.59**	164.64	**912.02**	0.48	984.14	0.52
inst29	40	979.70	265.95	1020.70	0.50	1050.08	0.53
inst30	40	962.11	281.35	1003.90	0.48	1052.76	0.51
inst31	50	1126.10	967.65	1220.60	0.55	1292.88	0.58
inst32	50	1217.20	350.73	1272.40	0.57	1330.74	0.59
inst33	50	1140.70	915.72	1174.60	0.56	1241.90	0.58
inst34	50	1175.70	866.13	1268.30	0.55	1319.93	0.58
inst35	50	1171.90	1130.57	1268.90	0.57	1318.17	0.58
inst36	50	1146.00	1280.97	1192.80	0.56	1236.52	0.58
inst37	50	1160.10	838.23	1251.60	0.56	1307.03	0.58
inst38	50	1199.90	1556.85	1294.60	0.56	1358.09	0.58
inst39	50	1219.60	1000.75	1299.10	0.56	1318.08	0.58
inst40	50	1129.20	875.78	1238.10	0.55	1283.75	0.59

company, as their network size can change from day-to-day and varies from city-to-city. It is also easily scalable and will achieve high-quality solutions in an on-demand setting.

Table 3: GCG result on Droppoint data

Trip	Route						
Trip01	CP	53	52	54	CP		
Trip02	CP	63	70	76	73	75	CP
Trip03	CP	69	67	66	64	65	CP
Trip04	CP	62	56	55	49	48	CP
Trip05	CP	6	5	4	2	1	CP
Trip06	CP	61	60	59	57	CP	
Trip07	CP	39	38	37	32	33	CP
Trip08	CP	16	8	7	3	15	CP
Trip09	CP	43	30	31	45	44	CP
Trip10	CP	58	51	50	46	40	CP
Trip11	CP	42	47	36	35	34	CP
Trip12	CP	41	12	10	9	14	CP
Trip13	CP	74	72	71	68	CP	
Trip14	CP	27	22	21	20	18	CP
Trip15	CP	28	26	25	24	23	CP
Trip16	CP	29	13	17	11	19	CP

6. Concluding Remarks

In this work, we have proposed a novel framework for solving a large-scale, on-demand spare-parts delivery problem. We developed (i) a hybrid solution approach utilizing the benefits of both column generation and metaheuristics, which is both (ii) is easily customizable, and capable of handling complex constraints, as well as adapting for a network that changes on a daily-basis. This hybrid approach enables the (iii) fast and efficient solution of a real-world network in under a few seconds, within 0.1%-9.6% of optimality (and an average 2.9%).

The proposed approach was tested on a real case study of spare parts distribution network in Sydney metropolitan area. The real case study involved a problem where a fleet of vehicles were used to timely deliver spare parts requested by maintenance technicians at various locations in and around Sydney metropolitan areas. The application highlighted the efficacy of proposed method for solving on-demand logistics optimization problems. Our solution procedure was also tested on a range of logistics network of varying sizes and complexity which highlighted the scalability of the solution procedure.
The problem formulation is based on the VRPTW and an intelligent combination of exact and approximate techniques was used as a solution technique. A hybrid column generation is employed as a method to enhance the overall efficiency and effectiveness of a classical Genetic Algorithm. The proposed approach reduces the economic and environmental impact in a lower computational time required for most on-demand type service provision. This work proposed has the potential to benefit hybrid metaheuristic strategies designed for combinatorial problems within the logistics setting and optimization in general.

Further performance improvements may be obtained in several ways. For instance, the idea of simultaneously running an evolutionary strategy from different initial points, on

several processors can considerably improve the explorative power of the search by a factor of number of threads. On the other hand, exchange of information between different threads can boost the exploitative power of the search procedure and reduce the required time to reach a high quality solution. It is this exploration/exploitation balance which can be exploited in the presented genetic column generation procedure, and problem specific exact heuristics, to play a key role in designing a parallel strategy. Other work on this topic can include i) the use of other more sophisticated evolutionary algorithms ii) testing proposed solution procedure in other outbound/inbound logistics scenarios and iii) parallel runs for the proposed solution procedure to gain computational efficiency.

The presented procedure was designed based on such exploitation-exploration trade-off to provide high quality solutions for the vehicle routing problem. The computational experiments indicated that this genetic integration is both effective and robust, producing high quality solutions in under one minute.

Acknowledgment

The authors thank DropPoint Australia Pty. Ltd for their collaboration and support.

References

Aggarwal, C. C., Orlin, J. B., Tai, R. P., 1997. Optimized crossover for the independent set problem. Operations Research 45 (2), 226–234.
URL http://www.jstor.org/stable/171740

Ahuja, R., Orlin, J., Tiwari, A., 2000. A greedy genetic algorithm for the quadratic assignment problem. Computers & Operations Research 27 (10), 917–934.

Alvelos, F., de Sousa, A., Santos, D., 2013. Combining column generation and metaheuristics. In: Talbi, E.-G. (Ed.), Hybrid Metaheuristics. Vol. 434 of Studies in Computational Intelligence. Springer Berlin Heidelberg, pp. 285–334.

Amirghasemi, M., Zamani, R., 2015. An effective asexual genetic algorithm for solving the job shop scheduling problem. Computers & Industrial Engineering 83, 123 – 138.
URL http://www.sciencedirect.com/science/article/pii/S0360835215000686

Amirghasemi, M., Zamani, R., Mar. 2017. An effective evolutionary hybrid for solving the permutation flowshop scheduling problem. Evolutionary Computation 25 (1), 87–111.
URL https://doi.org/10.1162/EVCO_a_00162

Baker, B. M., Ayechew, M., 2003. A genetic algorithm for the vehicle routing problem. Computers & Operations Research 30 (5), 787 – 800.

Baldacci, R., Bartolini, E., Mingozzi, A., 2011. An exact algorithm for the pickup and delivery problem with time windows. Operations Research 59 (2), 414–426.

Beasley, J. E., 1990. Or-library: Distributing test problems by electronic mail. The Journal of the Operational Research Society 41 (11), 1069–1072.
URL http://www.jstor.org/stable/2582903

Beck, J. C., Prosser, P., Selensky, E., 2002. On the reformulation of vehicle routing problems and scheduling problems. In: Abstraction, Reformulation, and Approximation. Springer, pp. 282–289.

Beheshti, A. K., Hejazi, S. R., 2015. A novel hybrid column generation-metaheuristic approach for the vehicle routing problem with general soft time window. Information Sciences 316, 598 – 615, nature-Inspired Algorithms for Large Scale Global Optimization.
URL http://www.sciencedirect.com/science/article/pii/S0020025514011189

Bierwirth, C., Mattfeld, D., Kopfer, H., 1996. On permutation representations for scheduling problems. Parallel Problem Solving from NaturePPSN IV, 310–318.

Borndörfer, R., Grötschel, M., Klostermeier, D., Küttner, C., 1999. Telebus Berlin: Vehicle Scheduling in a Dial-a-Ride System. In: Lecture Notes in Economics and Mathematical Systems. Proceedings of the 7th International Workshop on Computer-Aided Transit Scheduling. Springer, pp. 391–422.

Bräysy, O., Gendreau, M., 2005. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. Transportation Science 39 (1), 104–118.

Clerc, M., Kennedy, J., 2002. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary computation 6 (1), 58–73.

Cordeau, J.-F., Laporte, G., 2007. The Dial-a-Ride Problem: Models and Algorithms. Annals of Operations Research 153, 29–46.

Davis, L., 1985. Job shop scheduling with genetic algorithms. In: Grefenstette, J. (Ed.), Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ. pp. 136–140.

De Jong, K. A., 2006. Evolutionary computation: a unified approach. MIT Press.
URL http://books.google.com.au/books?id=OIRQAAAAMAAJ

Desaulniers, G., Erdmann, A., Solomon, M. M., Soumis, F., 2002. The VRP with Pickup and Delivery. The vehicle routing problem, SIAM Monographs on Discrete Mathematics and Applications. Philadelphia, pp. 225–242.

Desrosiers, J., Dumas, Y., Solomon, M. M., Soumis, F., 1995. Time Constrained Routing and Scheduling. Vol. 8 of Network Routing, Handbooks in Operations Research and Management Science. Elsevier Science, Amsterdam, pp. 35–139.

Djerid, L., Portmann, M., Villon, P., 1996. Performance analysis of permutation cross-over genetic operators. Journal of Decision Systems 4 (1/2), 157–177.

Dorigo, M., Gambardella, L. M., 1997. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1 (1), 53–66.

Eiben, A., Schippers, C., 1998. On evolutionary exploration and exploitation. Fundamenta Informaticae 35 (1), 35–50.

Elluru, S., Gupta, H., Kaur, H., Singh, S. P., Oct 2017. Proactive and reactive models for disaster resilient supply chain. Annals of Operations Research.
URL https://doi.org/10.1007/s10479-017-2681-2

Espinoza, D., Garcia, R., Goycoolea, M., Nemhauser, G. L., Savelsbergh, M. W. P., 2008. Per-Seat, On-Demand Air Transportation Part I: Problem Description and an Integer Multicommodity Flow Model. Transportation Science 42 (3), 263–278.

Falkenauer, E., Bouffouix, S., 1991. A genetic algorithm for job shop. In: IEEE International Conference on Robotics and Automation, 1991, Proceedings. pp. 824–829 vol.1.

Fox, B., McMahon, M., 1991. Genetic operators for sequencing problems. Foundations of genetic algorithms 1, 284–300.

Gao, L., Zhang, G., Zhang, L., Li, X., 2011. An efficient memetic algorithm for solving the job shop scheduling problem. Computers & Industrial Engineering 60 (4), 699 – 705.
URL http://www.sciencedirect.com/science/article/pii/S0360835211000143

Goldberg, D., Lingle, R., 1985. Alleles, loci and the traveling salesman problem. In: Proceedings of the 1st International Conference on Genetic Algorithms. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, pp. 154–159.

Holland, J. H., 1975. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press.

Iyer, S., Saxena, B., 2004. Improved genetic algorithm for the permutation flowshop scheduling problem. Computers & Operations Research 31 (4), 593–606.

Kaur, H., Singh, S. P., Dec 2016. Sustainable procurement and logistics for disaster resilient supply chain. Annals of Operations Research.
URL https://doi.org/10.1007/s10479-016-2374-2

Kaur, H., Singh, S. P., Feb 2017. Flexible dynamic sustainable procurement model. Annals of Operations Research.
URL https://doi.org/10.1007/s10479-017-2434-2

Moscato, P., 1989. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech Concurrent Computation Program, C3P Report 826, 1989.

Pacquette, J., Cordeau, J.-F., Laporte, G., Pascoal, M. M. B., 2013. Combining Multicriteria Analysis and Tabu Search for Dial-a-Ride Problems. Transportation Research Part B: Methodological 52, 1–16.

Parragh, S., 2011. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. Transportation Research Part C 19, 912–930.

Poon, P., Carter, J., 1995. Genetic algorithm crossover operators for ordering applications. Computers & Operations Research 22 (1), 135–147.

Potvin, J.-Y., Bengio, S., 1996. The vehicle routing problem with time windows part ii: Genetic search. INFORMS Journal on Computing 8 (2), 165–172.

Reeves, C., Yamada, T., 1998. Genetic algorithms, path relinking, and the flowshop sequencing problem. Evolutionary Computation 6 (1), 45–60.

Rochat, Y., Taillard, ., 1995. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1 (1), 147–167.
URL http://dx.doi.org/10.1007/BF02430370

Ropke, S., Cordeau, J.-F., Laporte, G., 2007. Models and branch-and-cut algorithms for pick up and delivery problems with time windows. Networks 49 (4), 258–272.

Shukla, N., Choudhary, A., Prakash, P., Fernandes, K., Tiwari, M., 2013a. Algorithm portfolios for logistics optimization considering stochastic demands and mobility allowance. International Journal of Production Economics 141 (1), 146 – 166.

Shukla, N., Dashora, Y., Tiwari, M. K., Chan, F. T. S., Wong, T. C., 2008. Introducing algorithm portfolios to a class of vehicle routing and scheduling problem. Proceedings of The 2nd International Conference on Operations and Supply Chain Management, 1 – 10.

Shukla, N., Kiridena, S., 2016. A fuzzy rough sets-based multi-agent analytics framework for dynamic supply chain configuration. International Journal of Production Research 54 (23), 6984–6996.
URL https://doi.org/10.1080/00207543.2016.1151567

Shukla, N., Tiwari, M., Ceglarek, D., 2013b. Genetic-algorithms-based algorithm portfolio for inventory routing problem with stochastic demand. International Journal of Production Research 51 (1), 118–137.

Solomon, M., 1983. Vehicle routing and scheduling with time window constraints: Models and algorithms. Ph.D. thesis, Dept. of Decision Sciences, University of Pennsylvania.

Solomon, M. M., 1987. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Operations Research 35 (2), 254–265.

Starkweather, T., Mcdaniel, S., Whitley, D., Mathias, K., 1991. A comparison of genetic sequencing operators. In: Proceedings of the fourth International Conference on Genetic Algorithms. Morgan Kaufmann, pp. 69–76.

Syswerda, G., 1991. Schedule optimization using genetic algorithms. Handbook of genetic algorithms, 332–349.

Toth, P., Vigo, D., 1997. Heuristic algorithms for the handicapped persons transportation problem. Transportation Science 31 (1), 60–71.

Wang, H., Lee, D.-H., Cheu, R., 2009. PDPTW Based Taxi Dispatch Modeling for Booking Service. In: Proceedings of the 5th International Conference on Natural Computation. ICNC'09. IEEE Press, Piscataway, NJ, USA, pp. 242–247.
URL http://dl.acm.org/citation.cfm?id=1797096.1797146

Whitley, L., Starkweather, T., Fuquay, D., 1989. Scheduling problems and traveling salesmen: The genetic edge recombination operator. In: Proceedings of the 3rd International Conference on Genetic Algorithms. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 133–140.