International Journal of Production Research

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

An integrated CPU-GPU heuristic inspired on variable neighbourhood search for the single vehicle routing problem with deliveries and selective pickups

I.M. Coelho[a], P.L.A. Munhoz[a], L.S. Ochi[a], M.J.F. Souza[b], C. Bentes[c] & R. Farias[d]

[a] Computer Science Institute, Fluminense Federal University, Niteroi, Brazil.

[b] Department of Computing, Federal University of Ouro Preto, Ouro Preto, Brazil.

[c] Department of System Engineering, State University of Rio de Janeiro, Rio de Janeiro, Brazil.

[d] COPPE-Systems Engineering, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil.

Published online: 27 Apr 2015.

To cite this article: I.M. Coelho, P.L.A. Munhoz, L.S. Ochi, M.J.F. Souza, C. Bentes & R. Farias (2015): An integrated CPU-GPU heuristic inspired on variable neighbourhood search for the single vehicle routing problem with deliveries and selective pickups, International Journal of Production Research, DOI: 10.1080/00207543.2015.1035811

To link to this article: http://dx.doi.org/10.1080/00207543.2015.1035811

International Journal of Production Research, 2015
http://dx.doi.org/10.1080/00207543.2015.1035811

An integrated CPU–GPU heuristic inspired on variable neighbourhood search for the single vehicle routing problem with deliveries and selective pickups

I.M. Coelho[a*], P.L.A. Munhoz[a], L.S. Ochi[a], M.J.F. Souza[b], C. Bentes[c] and R. Farias[d]

[a] *Computer Science Institute, Fluminense Federal University, Niteroi, Brazil;* [b] *Department of Computing, Federal University of Ouro Preto, Ouro Preto, Brazil;* [c] *Department of System Engineering, State University of Rio de Janeiro, Rio de Janeiro, Brazil;* [d] *COPPE-Systems Engineering, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil*

(*Received 4 March 2014; accepted 28 February 2015*)

Environmental issues have become increasingly important to industry and business in recent days. This trend forces the companies to take responsibility for product recovery, and proper recycling and disposal, moving towards the design of sustainable green supply chains. This paper addresses the backward stream in transportation of products, by means of reverse logistics applied to vehicle routing. This problem, called single vehicle routing problem with deliveries and selective pickups, consists in finding a route that starts from the depot and visits all delivery customers. Some pickup customers may also be visited, since the capacity of the truck is not exceeded, and there is also a revenue associated with each pickup. We develop an algorithm inspired on the variable neighbourhood search metaheuristic that explores the power of modern graphics processing unit (GPU) to provide routes in reasonable computational time. The proposed algorithm called four-neighbourhood variable neighbourhood search (FN-VNS) includes a novel high-quality initial solution generator, a CPU–GPU integrated perturbation strategy and four different neighbourhood searches implemented purely in GPU for the local search phase. Our experimental results show that FN-VNS is able to improve the quality of the solution for 51 instances out of 68 instances taken from the literature. Finally, we obtained speedups up to 14.49 times, varying from 17.42 up to 76.84 for each local search, measured over a set of new large-size instances.

Keywords: vehicle routing problems; reverse logistics; green supply chain management; GPU computing; variable neighbourhood search

1. Introduction

Environmental issues have received a great deal of attention over the past two decades. Organisations are adopting environmentally friendly management strategies in both operational and strategic contexts, aiming to add competitive advantages and to cope with environmental regulations (Sarkis and Rasheed 1995; Dalé et al. 2013). In addition, evidence of environmental damage intensifies the population pressure on organisations practices (Zhu and Sarkis 2007). In this scenario, green supply chain management (GSCM) is becoming an increasingly popular management concept because it takes environmental aspects into consideration when managing the supply chain (Srivastava 2007; Green et al. 2012; Validi, Bhattacharya, and Byrne 2014a, 2014b). Different research directions for addressing GSCM arise in: product design (Allenby 1993; Bras and McIntosh 1999), manufacturing practices (Winsemius and Guntram 1992; Gungor and Gupta 1999) and reverse logistics (Carter and Ellram 1998; Rogers and Tibben-Lembke 2001).

In this work, we focus on the reverse logistics aspect of GSCM (Van Hoek 1999). Reverse logistics deals with the backward movement, or the product return in the supply chain (De Brito and Dekker 2004). Developments in reverse logistics are fundamental for GSCM, providing waste reduction and cost savings, since a value can be retrieved from returned goods (Kumar, Teichman, and Timpernagel 2012). Environmental conscious organisations must take responsibility for their products at end of life, including product recovery and proper recycling and disposal. A considerable number of reverse logistics cases of study have been reported. A mail service problem is presented by Sural and Bookbinder (2003). The mail is stored in a depot and later on is delivered to the customers, which can also send mail back to the depot. Privé et al. (2005) present a practical application for the soft drink distribution problem, which includes the delivery of bottled water to customers, that return empty bottles back to the depot. The case of European electronics industry is studied in Nguyen (2012). Electrical and electronic equipments have a short life, and can be restocked and returned to the original

*Corresponding author. Email: imcoelho@ic.uff.br

distribution centres. The management of product return process in a timely and effective manner presented a great difficulty in Europe. Toktay, Wein, and Zenios (2000) present the case of single-use flash camera in Kodak. Customers take the used camera to a store that receives a deduction for each camera returned to the factory. The used circuit boards of cameras are further recycled to produce new cameras. The combined delivery and collection of products were studied in Bettac et al. (1999) for printer cartridge recycling in Great Britain, and power tools recycling in Germany. The recovery of copy machines is proposed by Thierry (1997), where used products are collected for recovery. Products are transported from plants to markets and from each market a certain amount of used products have to be collected. There are also some design considerations for product recovery networks in the literature (Fleischmann et al. 2000; Dekker and Fleischmann 2004).

In reverse logistics, the distribution and collection of the products have to be carefully planned. Transportation activities have to be organised in order to: (i) provide efficiency in the supply chain; (ii) reduce costs, guaranteeing the economic success of reprocessing products (Beullens, Van Oudheusden, and Van Wassenhove 2004); and (iii) avoid further degradations to the environment, endorsing the environmental benefits of the backward stream (Dethloff 2001). The dominant design decision in the transportation activities planning is the determination of the vehicle routes.

Vehicle routing is a well-known combinatorial optimization problem that aims to find vehicle routes at minimum cost. An important variant of this problem, called single vehicle routing problem with deliveries and selective pickups (SVRPDSP), fits the reverse logistics demands. This problem considers that a single vehicle delivers products to a set of customers, while some pickups are also possible during the route. Every satisfied pickup yields a positive revenue.

The SVRPDSP is a $\mathcal{NP}$-hard problem and this implies that there is no known polynomial algorithm that solves the problem to optimality. In this case, the use of metaheuristic frameworks and mathematical programming techniques can provide reasonably good solutions. However, they usually cannot handle efficiently large-size problems. In addition, online vehicle routing is gaining attention because of the advances in communication technologies. The demands of customers may change quickly and the manager may need a fast routing algorithm to solve the problem. Therefore, it is indisputable that companies need a computational mechanism to provide routes in reasonable computational time, despite the size of the problem.

Recent advances in processor architectures can be explored in order to accelerate routing algorithms. We present an efficient algorithm that explores the low-cost and highly accessible GPU technology. The GPUs have evolved into a powerful massively programmable parallel environment, where a large number of cores can be assembled for solving large-scale problems with lower price than multiple separated processing units. We propose an integrated heuristic, called FN-VNS, that combines the flexibility of the metaheuristic variable neighbourhood search (VNS) with acceleration components implemented in a GPU. The FN-VNS also includes a novel high-quality initial solution generator, integrated with two mathematical programming solvers, providing the search algorithm with good starting solutions. These solutions are further improved by local search in four different neighbourhoods that is parallelised by means of GPU programming. Finally, an integrated CPU–GPU perturbation mechanism and a novel stopping criteria introduce diversity during the search process and avoid premature convergence of the algorithm, preventing it from staying stuck in local optima. The results of FN-VNS are superior when compared to algorithms from literature and the integration with a GPU is able to achieve nearly 15 times speedup over a pure sequential version. It is also possible to achieve an acceleration of 76.84 times for the neighbourhood searches, considering a new set of instances of larger size.

In the next section, we briefly review works on strategic, tactical and operational decisions for supply chain management problems, focusing on successful applications of the VNS metaheuristic. We also review the works on GPU parallelisation for classic metaheuristic algorithms. In Section 3, we describe the problem in details, while Section 4 presents the proposed FN-VNS algorithm. The results are reported in Section 5. Finally, the Section 6 concludes the work and indicates directions for future researches in supply chain problems and parallel metaheuristics.

2. Related work

There are three important streams of research involving a supply chain management problem: strategic, tactical and operational (Farahani, Rezapour, and Kardar 2011). The strategic stream involves long-term decisions, such as facility location. The tactical stream involves decisions made in not such a long-term basis, such as transportation and fleet planning. The operational stream involves short-term decisions, usually made in real time, such as vehicle loading and route recomputation.

Facility location is a widely studied combinatorial optimization problem, that arises in industrial engineering, with applications in manufacturing units and process plants. It seeks an efficient location of facilities to optimise production flows and minimise the costs (Singh and Sharma 2006). This problem has been extensively studied since it was first formulated as a quadratic assignment problem (Koopmans and Beckmann 1957). Many heuristic techniques were developed since then

(Tompkins and Reed 1976; Hassan, Hogg, and Smith 1986) and some routing decisions were also included in the model, called the location-routing problem (LRP). It simultaneously determines the location of a set of facilities (depots) and optimises distribution in a supply chain where customers receive goods from the set of depots. Exact methods are usually based on a mathematical programming formulation (Laporte and Nobert 1981; Belenguer et al. 2011) and different heuristic approaches have been proposed to provide solutions in reasonable time (Hansen et al. 1994; Yu et al. 2010). Literature surveys on LRP are presented in Nagy and Salhi (2007) and Prodhon and Prins (2014) with the applications in logistics and distribution management.

From a tactic and operational perspective, in order to optimise transportation costs many practical applications are modelled as a vehicle routing problem (VRP). It has become a very important problem in supply chain logistics and also for computing, since it is an $\mathcal{NP}$-hard problem (Dantzig and Ramser 1959). Many heuristic algorithms have been developed for VRP variants (Toth and Vigo 2001; Golden, Raghavan, and Wasil 2008). The VNS framework proposed by Hansen, Mladenović, and Pérez (2010) has been applied to many hard combinatorial problems, including those related to the VRP. Currently, most of the best techniques for VRP variants are merged with multi-neighbourhood concepts inspired by the VNS, e.g. hybrid iterated local search (Subramanian, Uchoa, and Ochi 2013) and hybrid genetic algorithm (Vidal et al. 2013).

We discuss implementations that are based on VNS and parallel metaheuristics, as they have similarities with our approach. Recently, a VNS algorithm was successfully applied to an economic lot sizing problem with product returns and recovery (Sifaleras, Konstantaras, and Mladenovi 2015), indicating that this approach can also fit the GSCM objectives of this work. The work of Melechovskỳ, Prins, and Calvo (2005) deals with non-linear cost for the open depots, and presents a hybrid metaheuristic consisting of Tabu search and VNS. The work by Derbel et al. (2011) uses multiple neighbourhoods to evaluate solutions regarding both depot status and customer sequence. The perturbation mechanism is composed of two neighbourhoods affecting the depot, while the local search step uses classic neighbourhood structures for routing, inspiring the development of the proposed FN-VNS algorithm. In these works, the routing problem considers customers to have only delivery demands. The work of Ghodsi and Amiri (2010) considers pickup and delivery demands, but the pickups are not selective. A continuous location problem is solved and routing is solved with a VNS approach. In this case, a cross-exchange neighbourhood is used as perturbation and a 3-Opt operation as local search. Subramanian et al. (2010) developed a parallel iterated local search for a VRP with pickups and deliveries. The results indicate a tremendous decrease in the computational time using a massive number of processing cores. Noticeable speedups are achieved by Santos et al. (2010) for a greedy randomised adaptive search procedure metaheuristic, while some parallel evolutionary algorithms on the GPU were also proposed (Wong and Wong 2009; Luong et al. 2010). A Tabu search implementation with GPU is presented in Janiak, Janiak, and Lichtenstein (2008) and also in Luong et al. (2010) for a three-dimensional assignment problem. Finally, local search on the GPU is the subject of research of Luong, Melab, and Talbi (2013) and Schulz (2013).

The SVRPDSP was first presented by Sural and Bookbinder (2003). The authors contextualised the problem with some real industry cases and developed a mathematical programming model, using the MTZ sub-tour elimination constraints presented by Miller, Zemlin, and Tucker (1960). Experiments with the software CPLEX have shown that the proposed model was able to solve small randomly generated instances. However, the authors considered that customers have only delivery or pickup demands, but not both. This variant fits many real-world reverse logistic scenarios and it is closely related to the prize-collecting travelling salesman problem (PCTSP) presented by Balas (2004). The PCTSP consists in finding a tour visiting a subset of customers and penalising each non-visited customer. However, the PCTSP does not consider demands on customers, i.e. a scenario where the vehicle has unlimited capacity.

In Gribkovskaia, Laporte, and Shyshou (2008), a heuristic based on Tabu search was developed to solve the SVRPDSP. The idea behind the algorithm was to assign labels for each customer, *pickup only*, *delivery only*, *both pickup and delivery*, and exchange these labels in order to improve the solution quality. In terms of mathematical programming, a branch-and-cut algorithm for the SVRPDSP was developed by Gutiérrez-Jarpa, Marianov, and Obreque (2009). Efficient cuts were devised for a new mathematical programming model, which was able to solve some instances of up to 90 customers.

Coelho, Munhoz et al. (2012) proposed the first VNS with classic local search algorithms for the SVRPDSP. A basic initial solution was generated by randomly selecting pickup customers and adding random delivery customers in a route, merging them in a route. However, many infeasible routes may be generated with this strategy, when vehicle capacity is not respected. Also, the complicated perturbation scheme of the method involved an extra-perturbation parameter, causing computational times to rise much faster for larger instances of the problem. In Coelho, Ochi et al. (2012), an initial GPU parallelisation of local searches for the SVRPDSP was presented, achieving modest acceleration values. According to the authors, the lack of communication between the GPU components prevented the design of a fully integrated algorithm, so no improvement in the quality of solution was achieved by means of the GPU technology. Bruck, dos Santos, and Arroyo (2012) improved the perturbation mechanism using ideas from evolutionary algorithms while keeping the multi-neighbourhood characteristic of

VNS algorithms. The algorithm was able to improve marginally the quality of solutions and, to the best of our knowledge, these are the best results in literature for the problem at hand.

The proposed FN-VNS algorithm unifies good characteristics of recent VNS algorithms in literature, together with a novel solution generator with a global greedy criterion that merges two exact solutions from sub-problems of the SVRPDSP. This process is fast and allows the search to start from better quality initial solutions, further optimised by four different GPU local searches. The integrated perturbation scheme is a novel component that involves route recalculation in order to reduce communication time with the GPU and it is an important advance in order to design a complete metaheuristic in a CPU–GPU environment. From the managerial point of view, no parameter tuning is now involved with the proposed perturbation scheme in FN-VNS; thus, allowing the design of a fast and self-adaptable algorithm for small instances and even for instances with hundreds of customers.

3. The problem

In the SVRPDSP, the delivery demands of all customers must be satisfied, but the pickups are not obligatory. Consider a depot node c_0, a set of n customers $C_{\mathcal{D}} = \{c_1, c_2, \ldots, c_n\}$ and $C_{\mathcal{P}} = \{c_{n+1}, c_{n+2}, \ldots, c_N\}$ called *delivery customers* and *pickup customers*, respectively. The delivery and pickup customers have delivery demands d_i, pickups p_i and revenues r_i such that: $\forall c_i \in C_{\mathcal{D}}$, $d_i > 0$, $p_i = r_i = 0$; and $\forall c_{n+i} \in C_{\mathcal{P}}$, $d_{n+i} = 0$, $p_{n+i} > 0$, $r_{n+i} > 0$.

A solution S to the SVRPDSP consists of a permutation of all the delivery customers $C_{\mathcal{D}}$ and some selected pickup customers from $C_{\mathcal{P}}$. If a customer c_i precedes a customer c_j, then there is an arc connecting c_i to c_j with cost $M(c_i, c_j)$. A valid route starts and ends with the depot c_0 and respects the capacity Q of the vehicle, starting with load Q'. A load vector q is calculated such that $q_0 = Q'$ and for each customer c_i, a load of $q_i = q_{i-1} - d_i + p_i$, i.e. consider the delivery as a negative value and a pickup as a positive value.

Figure 1 depicts a solution to the problem with vehicle capacity $Q = 16$, five delivery and pickup customers, where positive values denote pickups and negative values denote deliveries. The depot (square) indicates the initial load at the vehicle ($Q' = 16$ for this example). Note that pickup customer +8 is not visited in this route, so no revenue is gained from this pickup.

The evaluation of a solution consists of the sum of transportation costs minus the sum of revenues associated with the pickups. In order to guide the search only for feasible routes, all load values q_i that exceed Q are multiplied by a huge constant H. The objective function is presented in Equation (1).

$$\min \sum_{(c_i, c_j) \in S} M(c_i, c_j) \; - \sum_{c_i \in S \,|\, c_i \in C_{\mathcal{P}}} r_i \; + \; H \times \sum_{c_i \in S} \max(q_i - Q, 0) \tag{1}$$

In Psaraftis (1983), an efficient technique is used in order to reduce the complexity of solution re-evaluations. We apply a similar technique for the feasible part of Equation (1), but due to the presence of multiple maximum functions in the infeasible part, the complexity of the re-evaluation is $\mathcal{O}(N)$. This is due to the fact that all maximum functions may change

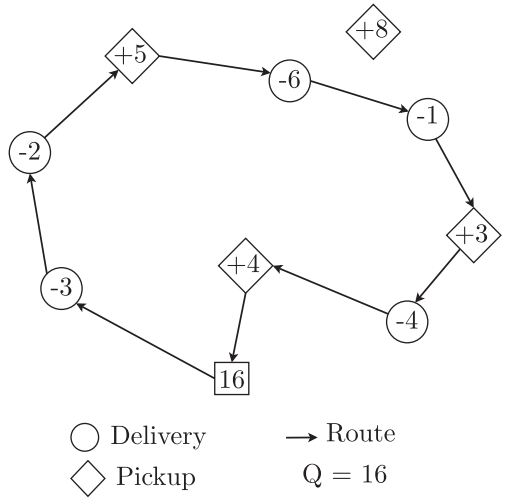

Figure 1. Solution $S = [depot, -3, -2, +5, -6, -1, +3, -4, +4, depot]$ to a SVRPDSP with five delivery and pickup customers. The depot is denoted by the square. Pickups are positive values and deliveries are negative.

their values in the worst case, so the number of recomputed values is linear. This fact motivated the implementation of the evaluation procedure in a GPU; thus, reducing the computational time using a big number of parallel processing units.

4. Four-neighbourhood variable neighbourhood search

The proposed FN-VNS algorithm relies on the basic principles of the metaheuristic framework VNS, also incorporating a new high-quality initial solution generator, an integrated CPU–GPU diversification mechanism and four different GPU searches.

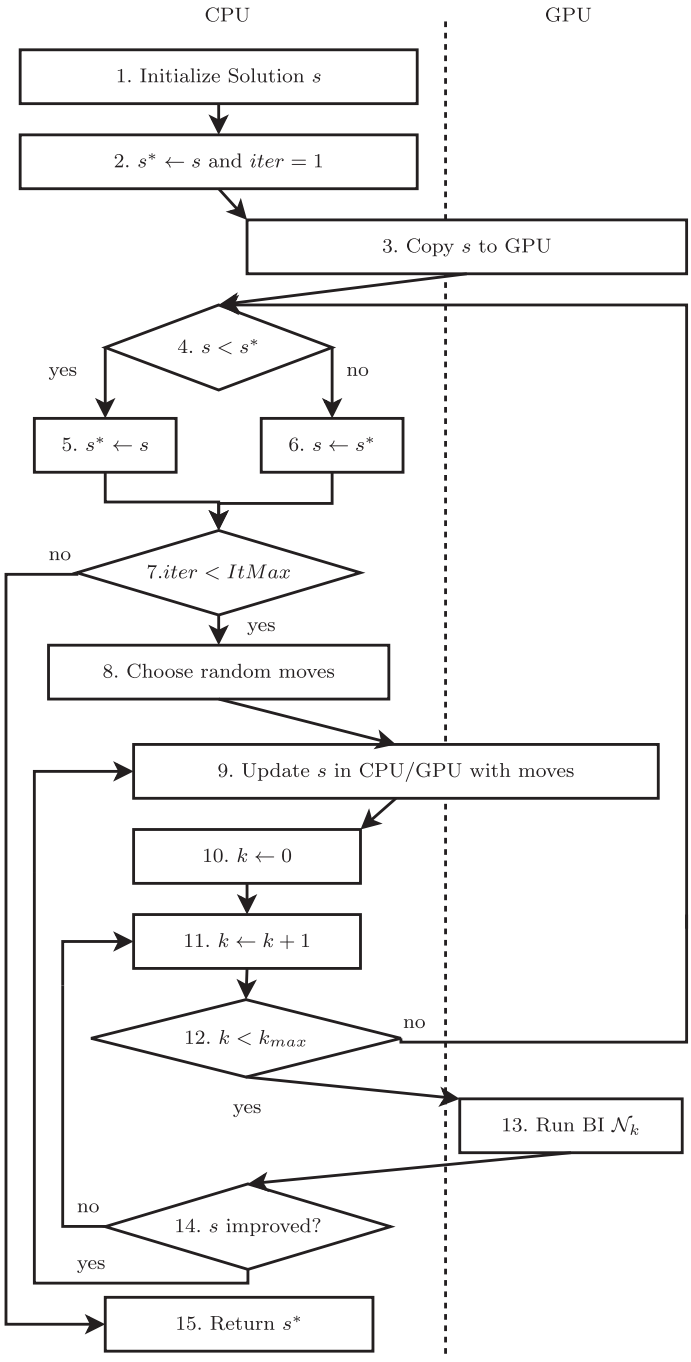

Figure 2. General framework of the developed FN-VNS.

The VNS is a metaheuristic framework proposed by Mladenović and Hansen (1997), which main idea is to apply a systematic change of some selected *neighbourhood structures* to explore the solution space. Let S be a solution to the optimization problem at hand and $\mathcal{N}_k(S)$ the set of neighbour solutions of S for neighbourhood k. A neighbour solution S' is reached from solution S by an operation called *move*. The neighbourhood can be fully explored in order to find better quality solutions. This classic heuristic, called best improvement (BI), is depicted in Algorithm 1.

Algorithm 1: Best Improvement (BI)

Input: S: Solution, $f(.)$: evaluation function, $\mathcal{N}_k(.)$: neighbourhood structure

1 $S' \leftarrow S$;
2 $S \leftarrow arg\ \min_{X \in \mathcal{N}_k(S)} f(X)$;
3 **if** ($f(S) \leq f(S')$) **then**
4 $\quad S' \leftarrow S$;
5 **end**
6 **return** S';

The schematic diagram for the FN-VNS is presented in Figure 2. The communications between CPU and GPU are represented in the diagram by boxes in the intersection between the CPU and GPU columns. For the manager that wishes to use only CPU technology, we provide also a fully CPU implementation of all GPU tasks. In this case, no communication between these hardwares is necessary. Steps 1–3 initialise the algorithm and copy the first generated solution to the GPU. The main loop is represented by steps 4–7, keeping the best known solution in s^* and limiting computational time to $IterMax$ iterations without improvement. In steps 8 and 9, the integrated perturbation generates random move operations and updates the solutions in both CPU and GPU. Steps 10–12 represent a classic variable neighbourhood descent (VND) strategy proposed by Hansen, Mladenović, and Pérez (2010), starting from the faster neighbourhoods ($k = 1$) and moving to the slower ones. In step 13, given a neighbourhood $\mathcal{N}_k$, the GPU finds the best neighbour with the `BI` algorithm. Step 14 checks if any improvement is made on the current solution and step 15 finishes the algorithm by returning the best solution s^*.

The FN-VNS algorithm starts with the generation of an initial solution for the SVRPDSP. In this algorithm, the problem is divided into two smaller sub-problems, tackled by two exact solvers and then merged together according to a greedy criterion. Let $S^*_{\mathcal{T}}$ be an optimal solution of value $f^*_{\mathcal{T}}$ to a travelling salesman problem consisting of all delivery customers plus the depot. Let $S^*_{\mathcal{K}}$ be an optimal solution of value $f^*_{\mathcal{K}}$ to a knapsack problem consisting of all pickup customers as items, maximising the revenue r_i of the customers subject to the vehicle capacity Q (Martello and Toth 1990). It is worth mentioning that $f^* = f^*_{\mathcal{T}} - f^*_{\mathcal{K}}$ is a lower bound for the SVRPDSP (Gribkovskaia, Laporte, and Shyshou 2008). The computation of the exact solutions $S^*_{\mathcal{T}}$ and $S^*_{\mathcal{K}}$ is also an $\mathcal{NP}$-hard problem, so approximations of these solutions can be done by means of heuristics when computational time is prohibitive for pure exact methods. Finally, the solutions $S^*_{\mathcal{T}}$ and $S^*_{\mathcal{K}}$ must be merged to form a valid SVRPDSP solution S. The solution starts with the arcs from the route $S^*_{\mathcal{T}}$. In the algorithm proposed by Coelho, Munhoz et al. (2012), the pickup customers are inserted in S in random positions. Since this process may lead to many infeasible solutions when the vehicle capacity is exceeded during the route, we propose a global greedy strategy based on best feasible insertions. In this strategy, every non-visited pickup customer from $S^*_{\mathcal{K}}$ is considered for an insertion in best position of the route, i.e. minimising the distance and maximising the revenue. The best insertion is chosen such that the

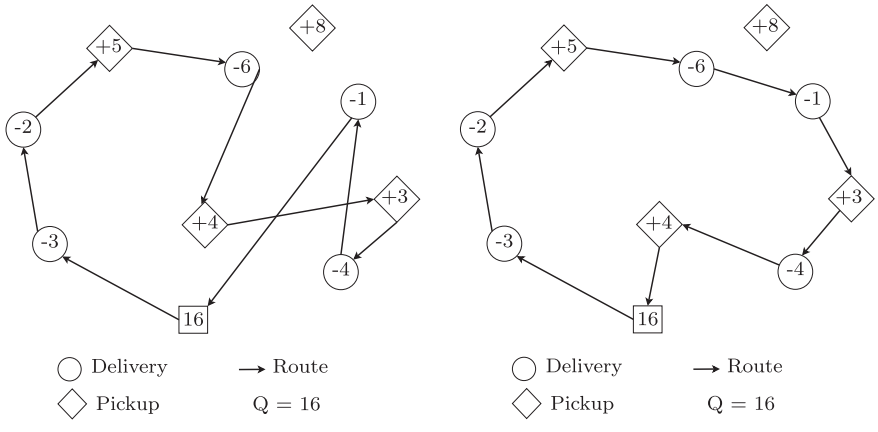

Figure 3. *Swap* exchanges customers +4 and −1, leading the solution $S = [depot, -3, -2, +5, -6, +4, +3, -4, -1, depot]$ to become the solution $S' = [depot, -3, -2, +5, -6, -1, +3, -4, +4, depot]$.

vehicle capacity is not violated and the process is repeated while there are pickup customers available. In the worst case, the complexity of the merge is $\mathcal{O}(N^2)$, since the route is gradually extended to $\mathcal{O}(N)$ customers with another linear calculation of $\mathcal{O}(N)$ at each iteration (for the best insertion and the feasibility test). Algorithm 2 describes this process in details.

The neighbourhood structures are the basis of a VNS algorithm, so the FN-VNS was designed with four different neighbourhood structures: *Swap*, *2-Opt*, *1-OrOpt* and *2-OrOpt*. Each of them is able to induce a different set of neighbour solutions, by means of different move operations.

Algorithm 2: Initial solution generation

Input: $C_{\mathcal{D}}$: Delivery customers, $C_{\mathcal{P}}$: pickup customers, M: cost matrix, Q: vehicle capacity, d: deliveries, p: pickups, r: revenues

$S^*_{\mathcal{T}} \leftarrow$ solve `TSP` $(C_{\mathcal{D}} \cup \{c_0\}, M)$;
$S^*_{\mathcal{K}} \leftarrow$ solve Knapsack Problem $(C_{\mathcal{P}}, Q, p, r)$;
$S \leftarrow S^*_{\mathcal{T}}$ `//Initialize solution S with TSP route;`
while $\exists\, c_i \in S^*_{\mathcal{K}} \mid c_i \notin S$ **do**
  $\Delta f^* \leftarrow 0$;
  for $c_i \in S^*_{\mathcal{K}} \mid c_i \notin S$ **do**
    Let Δf_i be the cheapest cost (distance) for the insertion of pickup customer c_i in S, respecting vehicle capacity Q;
    if $\Delta f_i - r_i < \Delta f^*$ **then**
      $\Delta f^* \leftarrow \Delta f_i - r_i$;
    end
  end
  Add pickup customer c_i to S according to best insertion Δf^*;
end
return S;

The *Swap* neighbourhood is a restriction of the Osman interchanges (Osman 1993) and consists of an exchange between two different customers. Figure 3 presents an example of a *Swap* move for the pickup customer $+4$ and delivery customer -1. It is worth mentioning that, in this case, the original solution (on the left of Figure 3, left) is not feasible due to the excess of one unit at customer $+3$. Indeed, considering $Q' = 16$, the load at customer $+3$ is: $Q' - 3 - 2 + 5 - 6 + 4 + 3 = 17 > Q = 16$. This situation is fixed after a swap operation in the solution (on the right of Figure 3).

The *2-Opt*, introduced by Lin and Kernighan (1973) for the TSP, removes two non-adjacent arcs from the route and adds two others generating a new route. In Figure 4, we show an example of a *2-Opt* where the arcs $(-1, +4)$ and $(+3, depot)$ are removed, while the new arcs $(-1, +3)$ and $(+4, depot)$ are created.

A *1-OrOpt* move (Or 1976) relocates one customer to another position in the route. The *2-OrOpt* is its natural extension when a block of two consecutive customers is relocated, instead of just one. Figure 5 presents an example of a *1-OrOpt* move where the delivery customer -6 is moved to the position immediately after pickup customer +5.

With a view to providing diversity during the search process, a novel perturbation approach is also proposed. While the search process can get stuck in a local optimum trap from a specific neighbourhood, a combination of moves from different

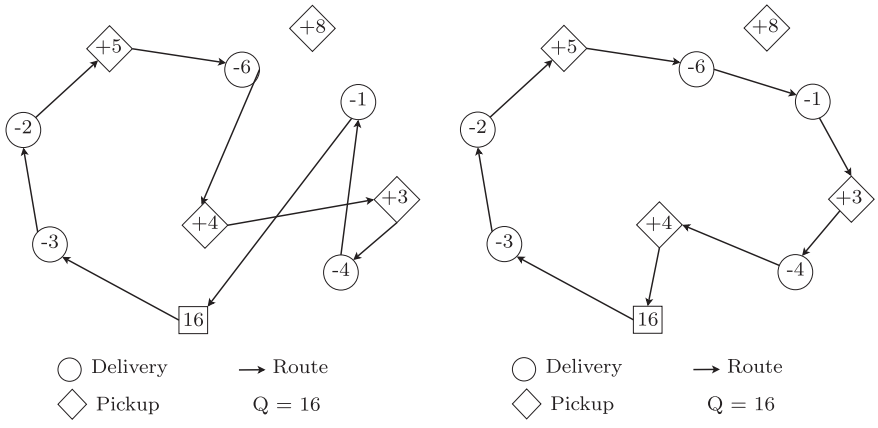

Figure 4. *2-Opt* removes arcs $(-1, +4)$ and $(+3, depot)$; and adds arcs $(-1, +3)$ and $(+4, depot)$, leading solution $S = [depot, -3, -2, +5, -6, -1, +4, -4, +3, depot]$ to $S' = [depot, -3, -2, +5, -6, -1, +3, -4, +4, depot]$.

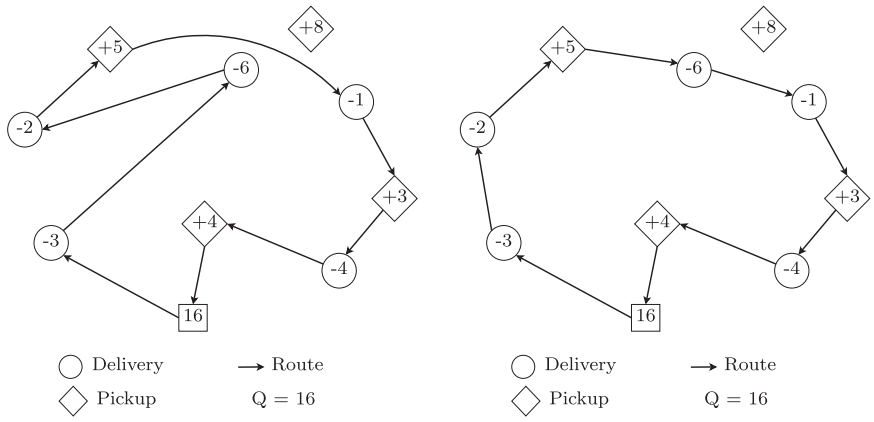

Figure 5. *1-OrOpt* moves customer -6 to the position after customer +5. The arcs $(-3,-6)$, $(-6,-2)$ and $(+5,-1)$ are removed, and the arcs $(-3,-2)$, $(+5,-6)$ and $(-6,-1)$ are created. Solution $S = [depot, -3, -6, -2, +5, -1, +3, -4, +4, depot]$ becomes $S' = [depot, -3, -2, +5, -6, -1, +3, -4, +4, depot]$.

neighbourhoods can be strong enough to drive the search process into a more promising region of the solution space. This diversification cannot be too weak, otherwise the search will fall again into the same local optimum, but if it is too strong, the search may skip a promising region. Due to the nature of the developed neighbourhood structures, the diversity procedure must perform at least two consecutive moves, what may be enough to scape from shallow local optima. But in order to scape from deeper local optima, the number of necessary consecutive moves can vary randomly, limited to a maximum of $\frac{|C_{\mathcal{D}}|+|C_{\mathcal{P}}|}{2}$ moves. Exceeding this limit would allow the solution to be completely rebuilt by the moves and the search process be driven to a random region in solution space, without the benefits of a high-quality initial solution.

The integrated CPU–GPU search method FN-VNS is presented in Algorithm 3.

Algorithm 3: FN-VNS Algorithm

Input: $iterMax$: max. number of iterations without improvement, $f(.)$: evaluation function, $\mathcal{N}_k(.)$: neighbourhoods, $C_{\mathcal{D}}$: delivery customers, $C_{\mathcal{P}}$: pickup customers, M: cost matrix, Q: vehicle capacity, d: deliveries, p: pickups, r: revenues

$S \leftarrow$ `InitialSolution`$(C_{\mathcal{D}}, C_{\mathcal{P}}, M, Q, d, p, r)$;
$iter \leftarrow 1$;
while $iter \leq iterMax$ **do**
  $S' \leftarrow S$;
  $l \leftarrow$ random number $\left[2, \frac{|C_{\mathcal{D}}|+|C_{\mathcal{P}}|}{2}\right]$;
  for $(i = 1$ ***to*** $l)$ **do**
    $k \leftarrow$ random number `[1, 4]`;
    $S'' \leftarrow$ random neighbor from $\mathcal{N}_k(S')$;
    $S' \leftarrow S''$;
  end
  for $(k = 1$ ***to*** $4)$ **do**
    $R \leftarrow$ `LocalSearch`$(S'', f(.), \mathcal{N}_k)$;
    $S^* \leftarrow$ `BI`$(R, S'', \mathcal{N}_k)$;
    if $(f(S^*) < f(S))$ **then**
      $S \leftarrow S^*$;
      $k \leftarrow 1$;
      $iter \leftarrow 0$;
    end
  end
  $iter \leftarrow iter + 1$;
end
return S;

The proposed neighbourhood structures are used in the diversification phase (lines 7 and 8) and also in the local search phase (line 12) in the following order (smaller to bigger neighbourhoods): *2-Opt*, *Swap*, *2-OrOpt* and *1-OrOpt*, defined as $\mathcal{N}_1$, $\mathcal{N}_2$, $\mathcal{N}_3$ and $\mathcal{N}_4$, respectively. This block of four neighbourhoods is denominated FN and gives the name to FN-VNS algorithm. The VND loop (lines 11–19) consists of: a `local search` procedure that explores the neighbourhood $\mathcal{N}_k$ in the GPU; a GPU feasibility test, that checks the viability of the solution; and a `BI` procedure, that updates the current solution. The feasibility test consists of checking every position in the solution vector if the capacity of the truck has been exceeded after a move. If an infeasible solution is found, the exceeding capacity is multiplied by a huge factor H, making the solution unattractive during the search process. The result of the local search process made by the GPU is stored in the auxiliary vector R. Later, if an improving move in found in R, the `BI` procedure applies the best move to the current solution. Whenever an improvement is made the $iter$ counter is set to zero and the neighbourhood counter k is also reset to the first neighbourhood. When no neighbourhood move is able to improve the current solution, the $iter$ counter is increased. FN-VNS stops when the $iter$ counter exceeds $iterMax$ iterations without improvement.

4.1 *Parallel local searches*

In metaheuristics, and particularly for the FN-VNS, the computational cost of the search process is strongly dominated by the local search (line 12 of Algorithm 3). This is explained by the complexity of the developed neighbourhood structures, which is $\mathcal{O}(N^3)$, since it is necessary to perform a $\mathcal{O}(N)$ feasibility test for each of the $\mathcal{O}(N^2)$ moves of each neighbourhood. In fact, the feasible part of the objective function in Equation (1) can be recalculated in constant time $\mathcal{O}(1)$, but the linearity of the feasibility test comes from the penalisation of infeasible solutions. In the worst case, all customers may exceed the vehicle capacity and every customer c_i may have a different penalised value $max(q_i - Q, 0) \times H$. Thus, we parallelised in a GPU this computationally demanding part of the code. The idea is to explore the massively parallel environment, the high memory bandwidth and the efficient thread scheduling mechanism of the GPU to reduce the computational times and to allow tackling larger problem instances.

The GPU implementation of a *Swap* move (i, j) is presented in Algorithm 4.

Algorithm 4: Kernel for *Swap* neighbourhood

Input: (i, j): thread indexes, S: Solution, q: Loads, R: result vector, N: number of customers, M: cost matrix, Q: vehicle capacity, d: deliveries, p: pickups, r: revenues

1 $value \leftarrow +M(S[i-1], S[j]) + M(S[j], S[i+1])$
2 $\quad +M(S[j-1], S[i]) + M(S[i], S[j+1])$
3 $\quad -M(S[i-1], S[i]) - M(S[i], S[i+1])$
4 $\quad -M(S[j-1], S[j]) - M(S_j, S_{j+1})$;
5 **foreach** *customer* $c_i \in S$ *in the position after the possible move* **do**
6 $\quad$ update evaluation with value $max(q_i - Q, 0) \times H$, if vehicle capacity is exceeded at customer c_i
7 **end**
8 $R[i \times N + j] \leftarrow value$;
9 **return** R;

The idea of the algorithm is to compute the cost of the move and to assert the feasibility of the new solution. As illustrated in Figure 6, the cost of the move is computed by replacing the distances from customers $i-1$ to i and i to $i+1$ and from customers $j-1$ to j and j to $j+1$, by the distances from customers $i-1$ to j and j to $i+1$ and from customers $j-1$ to i and i to $j+1$. The parallelisation technique can explore the best characteristics of the CPU and the GPU, by means of a massive computational GPU algorithm that returns a cost vector R to the main flow of FN-VNS running in the CPU.

5. Experimental results

We evaluate the performance and the quality of the solutions generated by FN-VNS. The results are compared against other SVRPDSP solutions in literature provided by Bruck, dos Santos, and Arroyo (2012). Experiments were performed on the 68 problem instances from Gribkovskaia, Laporte, and Shyshou (2008) and their respective lower bound values presented in literature. We divided the 68 instances from literature according to the instance sizes, *small*: 16–31 customers, *medium*: 33–72 customers, *large*: 76–101 customers. There are 28 instances in the *small* and *medium* groups; 12 instances in the *large* group. This new grouping scheme provides more insights on how FN-VNS deals with bigger problems.

Our experimental platform is composed by a CPU Intel i7 2.67 GHz, with 8 GB of memory (only one CPU core was used), and a GPU GeForce GTX 560 Ti, with 1 GB of memory and 384 cores. The proposed algorithm was implemented

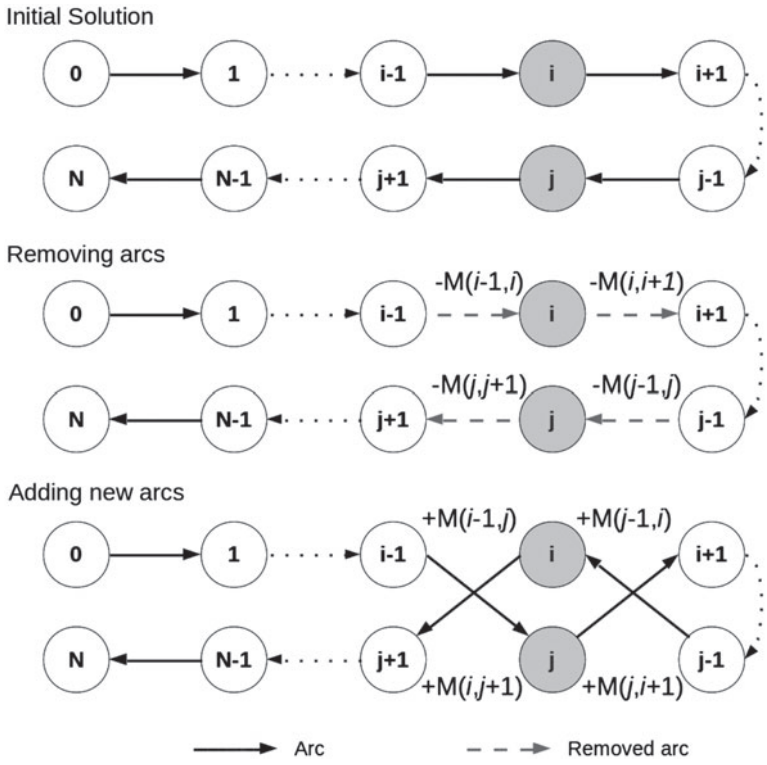

Figure 6. *Swap* exchanges customers i and j by removing arcs $i-1$ to i, i to $i+1$, $j-1$ to j, j to $j+1$, and adding arcs $i-1$ to j, j to $i+1$, $j-1$ to i, i to $j+1$.

Table 1. Experiments with initial solution generation.

Group of	Coelho, Munhoz et al. (2012)		Proposed constructive	
instances	gap(%)	time(ms)	gap(%)	time(ms)
small	655.61	0.007	142.64	1.606
medium	696.84	0.012	172.83	19.601
large	2286.46	0.019	589.29	192.711
Average	962.87	0.011	233.03	42.917

in C++ and CUDA version 4.0. After preliminary tests, we fixed the algorithm to 256 threads per block and also limited the registers per block to 20, in order to achieve a better occupancy of the GPU processors.

5.1 *Quality of initial solution*

In order to test the quality of the solutions generated by the proposed global greedy constructive (Algorithm 2), it was compared to the randomised merge algorithm of Coelho, Munhoz et al. (2012). Table 1 presents the results in terms of gap and $time$. The gap is calculated in relation to the lower bound for each instance, i.e. $gap(value) = (value - lower\ bound)/|lower\ bound|$. The column $time$ presents the execution times in milliseconds.

From Table 1, it is possible to see that the proposed global greedy strategy generates solutions with much lower gaps than a pure random generation. On average, a gap over 900% is obtained from a random merge, while gaps of 233% are achieved by a more elaborated greedy strategy. The running times of the proposed method are bigger, but less than one second on average, so they are still very small compared to the execution times of the complete FN-VNS.

Table 2. Comparison of solution quality after local searches (including *3-Opt*).

Group of instances	Improvement(%)					
	2-Opt	*Swap*	*2-OrOpt*	*1-OrOpt*	*FN*	*3-Opt*
small	18.0	23.7	20.4	27.1	54.2	30.6
medium	28.0	39.2	38.9	42.1	131.3	42.8
large	4.3	6.3	5.8	6.2	20.5	7.1
All	19.7	27.0	25.4	29.6	80.0	31.5

Table 3. Comparison of computational times of local searches (including *3-Opt*).

Group of instances	Time(ms)					
	2-Opt	*Swap*	*2-OrOpt*	*1-OrOpt*	*FN*	*3-Opt*
small	1.8	2.0	4.6	5.3	27.5	125.5
medium	13.0	13.5	32.1	34.1	181.5	1891.5
large	79.4	82.3	191.1	199.5	1067.9	18596.0
All	20.1	20.9	48.9	51.4	274.5	4112.2

5.2 *Experiments with local searches*

Since the neighbourhoods *1-OrOpt* and *2-OrOpt* are special cases of the classic neighbourhood *3-Opt*, we investigated the impact of this neighbourhood for the SVRPDSP. Table 2 presents the improvement of each local search after the constructive method. Considering all instances, the *2-Opt* improves only 19.7% on average, while the *3-Opt* produces the best results for single local searches, with 31.5% of improvement on average. However, the FN structure consisting of neighbourhoods *2-Opt*, *Swap*, *2-OrOpt* and *1-OrOpt* explored as presented in Algorithm 3, lines 11–19, is capable of an improvement on initial solutions of 80.0% on average.

Table 3 presents the computational time of each local search after the constructive method. From this table, it is possible to see that neighbourhoods *2-Opt*, *Swap*, *2-OrOpt* and *1-OrOpt* are ordered in terms of computational time, from the least to the most expensive. The search process of the four neighbourhoods FN combined take 274.5 ms on average, while the *3-Opt* takes nearly 15 times more time, 4112.2 ms. This is due to the much bigger number of elements in the *3-Opt* neighbourhood, which is $\mathcal{O}(N^3)$, compared to the others that are quadratic. This experiment motivated us to discard the *3-Opt* in our final algorithm, since the FN structure alone is already capable of big improvements, while retaining lower computational times.

5.3 *Comparison with literature*

The execution time of FN-VNS depends on the parameter $iterMax$, which was set to 200 after a preliminary battery of tests. The calibration of this parameter seeks a good compromise between quality of solution and execution times. Since the loop variable $iter$ is reinitialised after every improvement, the execution time grows quickly depending on the number of improvements and the shape of the solution space. This behaviour is not linear and should be experimented empirically depending on the strength of the perturbation and local searches involved. The constant H was also calibrated and set to 100,000. This decision considered programming language details, e.g. limitations on floating point types in C++ with very large values, also empirical tests were done to guarantee that no infeasible solution is chosen if it is possible to reach another high-value feasible solution.

Tables 4 and 5 show FN-VNS results for the 68 instances from literature and more 32 new instances generated by the authors,[1] considering from 150 up to 484 customers. In order to assess the quality of the solutions produced by FN-VNS, these are compared to a lower bound of the problem proposed by Gribkovskaia, Laporte, and Shyshou (2008), which is presented in column `LB`. Column `EA` presents the best solutions found by the evolutionary algorithm of Bruck, dos Santos, and Arroyo (2012). Columns `best` and `avg` present, respectively, the best and average solutions found by FN-VNS in 10 executions. Column `gap` shows the gap between the best solution found by FN-VNS and the lower bound of the problem. Column `var` shows the variability of the average solutions in relation to the best solutions found by FN-VNS, given by:

Table 4. Results for the 68 instances from literature (Part I).

Instance	LB	EA	best	avg	gap(%)	var(%)	time(s)$_{\times spd}$
016_B_half	36.60	**36.60**	**36.60**	39.21	0.00	7.14	$1.91_{\times 1.03}$
016_B_one	−155.41	**−150.73**	**−150.73**	−150.73	3.01	0.00	$1.33_{\times 1.06}$
016_B_p_two	130.99	135.11	**132.27**	134.26	0.98	1.50	$2.45_{\times 0.93}$
016_B_two	−540.81	**−536.13**	**−536.13**	−536.13	0.87	0.00	$1.35_{\times 1.09}$
021_B_half	−20.16	−20.16	−18.62	−12.56	7.64	32.52	$2.78_{\times 1.77}$
021_B_one	−316.09	−301.93	**−307.79**	−307.79	2.63	0.00	$2.26_{\times 1.77}$
021_B_p_two	132.21	146.75	**137.59**	141.37	4.07	2.75	$4.03_{\times 1.69}$
021_B_two	−909.57	**−901.28**	**−901.28**	−901.28	0.91	0.00	$2.56_{\times 1.77}$
022_B_half	−64.97	**−62.63**	**−62.63**	−58.01	3.60	7.37	$2.85_{\times 1.93}$
022_B_one	−429.15	−421.04	**−421.03**	−421.03	1.89	0.00	$2.47_{\times 1.93}$
022_B_p_two	116.01	124.25	**123.60**	124.17	6.54	0.46	$2.50_{\times 1.75}$
022_B_two	−1157.49	**−1149.38**	**−1149.38**	−1149.38	0.70	0.00	$2.47_{\times 1.94}$
023_B_half	−94.06	**−80.95**	**−80.95**	−80.95	13.94	0.00	$2.75_{\times 2.05}$
023_B_one	−711.46	**−698.35**	**−698.35**	−698.35	1.84	0.00	$2.74_{\times 2.06}$
023_B_p_two	260.88	274.31	**269.12**	269.82	3.16	0.26	$2.80_{\times 1.98}$
023_B_two	−1946.21	**−1933.10**	**−1933.10**	−1933.10	0.67	0.00	$2.75_{\times 2.05}$
026_B_half	−92.47	−91.95	−88.20	−82.92	4.62	5.98	$5.79_{\times 2.54}$
026_B_one	−504.40	**−497.17**	**−497.17**	−497.17	1.44	0.00	$2.96_{\times 2.63}$
026_B_p_two	139.67	146.51	148.03	150.31	5.99	1.54	$4.55_{\times 2.53}$
026_B_two	−1341.49	**−1334.26**	**−1334.26**	−1334.26	0.54	0.00	$2.97_{\times 2.61}$
030_B_half	−382.80	−357.21	**−377.12**	−375.73	1.48	0.37	$8.47_{\times 3.63}$
030_B_one	−1156.23	−1120.33	**−1150.44**	−1149.35	0.50	0.09	$8.75_{\times 3.63}$
030_B_p_two	81.33	**82.29**	**82.29**	86.27	1.19	4.82	$8.11_{\times 3.57}$
030_B_two	−2703.16	−2667.25	**−2697.37**	−2696.28	0.21	0.04	$8.76_{\times 3.65}$
031_B_half	−91.79	−85.56	**−89.32**	−87.15	2.69	2.43	$6.62_{\times 3.65}$
031_B_one	−514.05	−505.46	**−510.50**	−509.25	0.69	0.24	$7.26_{\times 3.60}$
031_B_p_two	115.52	**123.15**	**123.15**	125.34	6.61	1.77	$8.16_{\times 3.37}$
031_B_two	−1358.57	−1350.61	**−1355.02**	−1353.77	0.26	0.09	$7.27_{\times 3.61}$
033_B_half	−157.09	−137.29	**−144.66**	−144.66	7.91	0.00	$5.57_{\times 4.06}$
033_B_one	−778.21	−759.25	**−765.76**	−765.76	1.60	0.00	$5.50_{\times 4.07}$
033_B_p_two	188.44	198.44	**197.01**	203.55	4.55	3.32	$10.73_{\times 4.00}$
033_B_two	−2020.40	−2001.44	**−2007.89**	−2007.89	0.62	0.00	$5.87_{\times 4.08}$
036_B_half	−128.53	−113.44	**−128.25**	−126.20	0.22	1.60	$11.08_{\times 4.79}$
036_B_one	−624.67	−608.73	**−624.41**	−622.59	0.04	0.29	$10.99_{\times 4.79}$
036_B_p_two	121.94	139.71	**132.05**	134.52	8.29	1.87	$9.27_{\times 4.19}$
036_B_two	−1616.90	−1596.84	**−1616.63**	−1614.81	0.02	0.11	$11.19_{\times 4.73}$
041_B_half	−186.35	−184.07	−183.46	−178.00	1.55	2.97	$13.67_{\times 5.67}$
041_B_one	−767.97	−755.69	**−760.53**	−758.88	0.97	0.22	$13.88_{\times 5.82}$
041_B_p_two	100.89	111.66	**110.68**	114.93	9.70	3.84	$11.20_{\times 5.29}$
041_B_two	−1931.31	−1922.74	**−1923.88**	−1922.23	0.38	0.09	$13.83_{\times 5.87}$

$$var_i = \frac{avg_i - best_i}{|best_i|} \tag{2}$$

Finally, column time(s)$_{\times spd}$ presents the total time (in seconds) spent by the parallel FN-VNS and the speedup in relation to the sequential FN-VNS, following the same notation of Luong, Melab, and Talbi (2013). The CPU times are not presented in the tables since they can be easily deduced. Values printed in bold face show the improvements or the ties of the proposed algorithm upon the work by Bruck, dos Santos, and Arroyo (2012). Unfortunately, it is not possible to compare the solution quality with the work of Gribkovskaia, Laporte, and Shyshou (2008) for each instance, since in this work, only general-quality results are presented.

Our algorithm was able to find 51 new better solutions and 7 equal solutions, out of the 68 instances used. The average gap of 5.32% is also quiet low, since we compared the solutions with a lower bound of the problem. This shows that the best solutions achieved by the proposed algorithm are near optimal solutions. Finally, the low variability of 1.55% from the average solutions shows that the proposed algorithm is robust.

Table 5. Results for the 68 instances from literature (Part II).

Instance	LB	EA	best	avg	gap(%)	var(%)	time(s) $_{\times spd}$
045_B_half	−491.15	−489.47	**−491.15**	−491.15	0.00	0.00	$0.72_{\times 6.48}$
045_B_one	−1648.51	−1647.59	**−1648.51**	−1648.51	0.00	0.00	$0.73_{\times 6.40}$
045_B_p_two	198.04	202.70	**199.84**	202.18	0.91	1.17	$12.60_{\times 6.62}$
045_B_two	−3963.32	**−3963.32**	**−3963.32**	−3963.32	0.00	0.00	$0.73_{\times 6.33}$
048_B_half	−37752.96	**−36786.80**	**−36786.64**	−36705.55	2.56	0.22	$20.42_{\times 6.96}$
048_B_one	−107058.78	−105863.00	**−106625.21**	−106542.29	0.40	0.08	$17.62_{\times 7.14}$
048_B_p_two	−4797.03	−3830.83	**−4244.72**	−3802.70	11.51	10.41	$18.36_{\times 6.98}$
048_B_two	−248298.30	−247332.00	**−247864.73**	−247781.81	0.17	0.03	$17.59_{\times 7.15}$
051_B_half	−320.61	−310.02	**−311.26**	−309.53	2.92	0.56	$21.28_{\times 7.94}$
051_B_one	−1098.86	−1083.76	**−1089.51**	−1087.78	0.85	0.16	$21.28_{\times 7.93}$
051_B_p_two	116.58	135.34	**130.99**	133.85	12.36	2.18	$21.58_{\times 7.17}$
051_B_two	−2655.30	−2640.20	**−2645.94**	−2644.20	0.35	0.07	$21.27_{\times 7.92}$
072_B_half	−409.78	−388.84	**−407.76**	−405.92	0.49	0.45	$67.04_{\times 12.13}$
072_B_one	−1027.24	−998.14	**−1024.53**	−1023.20	0.26	0.13	$79.74_{\times 12.20}$
072_B_p_two	−39.24	−19.05	**−36.20**	−35.41	7.75	2.18	$58.55_{\times 12.22}$
072_B_two	−2262.21	−2232.95	**−2259.32**	−2257.63	0.13	0.07	$80.24_{\times 12.16}$
076_B_half	−579.52	−565.60	**−574.08**	−570.02	0.94	0.71	$66.57_{\times 12.15}$
076_B_one	−1759.19	−1741.31	**−1754.68**	−1749.45	0.26	0.30	$79.42_{\times 12.17}$
076_B_p_two	14.33	34.04	38.41	38.41	168.04	0.00	$35.58_{\times 10.87}$
076_B_two	−4118.50	−4099.12	**−4114.30**	−4108.92	0.10	0.13	$73.75_{\times 12.29}$
101a_B_half	−922.71	−911.68	−903.12	−896.57	2.12	0.73	$221.25_{\times 14.08}$
101a_B_one	−2552.38	−2538.99	−2531.06	−2525.65	0.84	0.21	$186.36_{\times 13.91}$
101a_B_p_two	−66.59	−47.99	−46.79	−46.79	29.73	0.00	$99.51_{\times 12.19}$
101a_B_two	−5811.69	−5800.66	−5790.86	−5785.43	0.36	0.09	$170.43_{\times 13.50}$
101b_B_half	−1386.67	−1380.63	**−1381.01**	−1379.78	0.41	0.09	$197.91_{\times 14.49}$
101b_B_one	−3320.83	−3314.79	**−3315.35**	−3312.92	0.17	0.07	$187.62_{\times 13.59}$
101b_B_p_two	−257.18	−251.14	−248.74	−245.12	3.28	1.46	$181.00_{\times 13.07}$
101b_B_two	−7188.92	−7182.88	**−7183.25**	−7181.20	0.08	0.03	$175.00_{\times 13.88}$
Average					5.32	1.55	$34.95_{\times 6.08}$

5.4 *Efficiency of GPU acceleration*

As can be observed in Tables 4 and 5, FN-VNS improved the execution time for all the instances tested, except for one. The speedups increase with the problem size, varying from 0.93 to 14.49. So, for smaller instances, the GPU time is nearly the same as the CPU time, but it is almost 15 times faster for the larger instances, showing the capability of the parallel FN-VNS to deal with larger problems.

Despite the great speedups obtained by FN-VNS, it is important to assess the efficiency of the local search separately. To perform this evaluation, we further included 32 new instances as a group *huge*, that we generated according to the methodology proposed in literature Gribkovskaia, Laporte, and Shyshou (2008).

Table 6 shows the average and maximum speedups (`avg`, `max`) obtained by each neighbourhood of the parallel FN-VNS algorithm when compared to the sequential execution of the neighbourhood for each of the group of instances. The results show that our four proposed GPU neighbourhoods consistently outperform the CPU counterparts, mainly for the larger instances where there is enough work to keep the threads busy in the highly parallel architecture of the GPU.

Analysing the larger instances results in detail, we can observe that the *Swap* neighbourhood presents the worst average speedup for the *large* instances. The *Swap* neighbourhood handles eight arcs in the route, while *k-OrOpts* deal with six arcs and *2-Opt* deals with four arcs. This makes *Swap* the most expensive neighbourhood in terms of computation and memory accesses. The necessary accesses to the global memory to update the solution vector `S`, make *Swap* slower than the others. However, the results are different for the *huge* instances. It seems that the cache memory in GPU global memory works better for larger routes, where the accesses of threads in closer blocks are stored more efficiently by the cache controller, increasing the speed of the access.

To validate the results obtained for this new set of instances, we evaluate the quality of the results obtained for these instances. Table 7 presents the lower bound (`LB`), the best and average solutions (`best, avg`) found by FN-VNS, the gap (`gap`), variability (`var`) and execution time in seconds (`time`). Due to the greater number of customers, the parameter

Table 6. Speedup results for FN-VNS neighbourhoods.

Group of instances	Number of instances	*Swap*		*2-Opt*		*1-OrOpt*		*2-OrOpt*	
		avg	max	avg	max	avg	max	avg	max
small	28	1.81	3.33	2.61	4.73	3.61	6.21	3.16	6.14
medium	28	7.63	16.41	10.36	20.49	12.81	24.69	11.28	21.00
large	12	17.42	19.21	24.47	29.45	26.50	28.97	24.49	26.42
huge	32	55.56	76.84	45.74	52.83	58.36	71.95	54.23	64.79
All	100	22.51		21.21		26.45		24.33	

Table 7. Quality of the solution for the generated *huge* instances.

Instance	LB	best	avg	gap(%)	var (%)	time(s)
151_B_half	−1640.84	−1618.16	−1615.62	1.38	0.16	40.12
151_B_one	−4067.39	−4042.81	−4036.56	0.60	0.15	48.97
151_B_p_two	−267.52	−245.54	−236.71	8.22	3.60	34.35
151_B_two	−8920.43	−8903.88	−8890.57	0.19	0.15	51.96
200_B_half	−2259.56	−2231.61	−2224.00	1.24	0.34	122.76
200_B_one	−5402.65	−5378.39	−5361.71	0.45	0.31	143.94
200_B_p_two	−512.95	−492.01	−489.65	4.08	0.48	98.29
200_B_two	−11688.67	−11660.55	−11645.60	0.24	0.13	156.08
256_B_half	−1016.09	−1007.79	−1006.63	0.82	0.12	377.10
256_B_one	−2459.00	−2452.99	−2449.50	0.24	0.14	308.65
256_B_p_two	−151.77	−145.26	−142.39	4.29	1.98	390.21
256_B_two	−5343.33	−5335.78	−5333.55	0.14	0.04	363.09
321_B_half	−2031.52	−2027.10	−2021.58	0.22	0.27	522.61
321_B_one	−4902.92	−4900.16	−4893.26	0.06	0.14	508.53
321_B_p_two	−410.10	−403.44	−399.92	1.62	0.87	372.66
321_B_two	−10645.99	−10637.42	−10633.76	0.08	0.03	526.59
386_B_half	−46685.80	−46685.80	−46563.66	0.00	0.26	507.64
386_B_one	−102744.33	−102744.33	−102591.27	0.00	0.15	507.56
386_B_p_two	−13050.74	−13050.74	−12999.88	0.00	0.39	461.56
386_B_two	−214862.18	−214862.18	−214695.98	0.00	0.08	461.55
400_B_half	−2164.30	−2155.26	−2152.81	0.42	0.11	1025.44
400_B_one	−4999.38	−4989.82	−4986.61	0.19	0.06	1083.17
400_B_p_two	−463.18	−455.22	−452.02	1.72	0.70	1003.52
400_B_two	−10669.60	−10662.82	−10658.66	0.06	0.04	1051.67
481_B_half	−3988.65	−3975.52	−3967.16	0.33	0.21	2283.14
481_B_one	−9253.49	−9244.92	−9233.44	0.09	0.12	2576.68
481_B_p_two	−845.94	−840.97	−833.26	0.59	0.92	1512.03
481_B_two	−19782.77	−19768.90	−19762.41	0.07	0.03	2573.12
484_B_half	−2955.05	−2944.73	−2941.37	0.35	0.11	2181.63
484_B_one	−6722.59	−6712.73	−6708.17	0.15	0.07	2181.42
484_B_p_two	−694.69	−686.83	−681.94	1.13	0.71	2438.10
484_B_two	−14256.85	−14248.10	−14243.64	0.06	0.03	2343.49
Average				0.91	0.40	883.05

$iterMax$ of the algorithm was reduced to 20, but even with this small number of iterations, the sequential algorithm was not able to finish in reasonable times, so only the parallel FN-VNS results are presented.

As can be seen in table, even for such large instances, the parallel FN-VNS algorithm succeeded in finding near optimal solutions, with a gap of 0.91% on the average, and also with a low variability of 0.40% from the average solutions. This validates the robustness of our approach. It is important to notice that the instances in the *huge* group were created for this work, making it impossible to compare their results with other authors.

Table 8. Performance gains of `Shared-S` implementation over `Global-S` implementation.

Group of instances	Number of instances	Average time decrease			
		Swap	*2-Opt*	*1-OrOpt*	*2-OrOpt*
*small**	28	–	–	–	–
*medium**	28	–	–	–	–
*large**	12	–	–	–	–
huge	32	3.12%	3.15%	3.32%	6.26%

*In groups *small*, *medium* and *large*, no statistical differences were observed with 95% confidence in a non-parametric test.

5.5 *Memory hierarchy*

The parallelisation of the local search of FN-VNS exploits the massive computational capacity of the GPU. Nevertheless, there is still one important issue that needs to be investigated, how to deal with the GPU memory hierarchy efficiently. In terms of memory hierarchy, GPUs provides different memories with different bandwidths that can be leveraged to improve performance.

The solution vector S has a linear size and we implemented two different allocation schemes for S. In the first one, called `Global-S`, we allocate S on the large high latency global memory and in the second one, called `Shared-S`, we allocate S in the low latency on-chip shared memory. The vector R was allocated in global memory due to its quadratic size.

We also made experiments to evaluate the impact of exploring the memory hierarchy of the GPU. In these experiments, we compare our two different allocations of the solution vector, `Global-S` and `Shared-S`. Table 8 shows the percentage of the performance gains in the execution times for the four neighbourhoods of the `Shared-S` implementation over the `Global-S` implementation, for each group of instances. As can be observed in this table, for the *small*, *medium* and *large* groups, the usage of the shared memory provides no substantial improvement. This occurs because the L2 cache mechanism of the GPU is effective in handling the accesses to the solution vector. In the *huge* group, the usage of the shared memory improves the execution time of the neighbourhoods from 3.12 to 6.26%, on the average. This improvement is achieved because in such large instances, the solution vector does not fit into the L2 cache. The misses in the L2 cache, which require accesses to the global memory, make the `Global-S` slower than the `Shared-S`. The gains of `Shared-S` over `Global-S` are, however, modest. In the `Shared-S` implementation, all threads in a block must copy collaboratively the vector S to the shared memory. After the vector is copied, it is accessed with a very low latency, but only a few elements are updated. So, the ratio between the cost of copying the data and the data reuse is high, what explains the small gains obtained.

6. Conclusions

In this work, we have dealt with the SVRPDSP. It is a very important and practical problem arising in reverse logistics and also in the design of green supply chains. The selective pickup component of the problem allows to include in the vehicle routing the product recovery, proper recycling and disposal. So, besides the optimisation of a vehicle fleet for product deliveries, it provides waste reduction and cost savings by means of the value retrieved from returned goods. Many real-world cases illustrate the need of a selective pickup component, such as soft drink distribution and distribution/recovery of copy machines.

Since the computation of the vehicle route is a $\mathcal{NP}$-hard problem, an efficient mechanism to provide routes in reasonable computational time is determinant for the success of the industry/business transportation problem. To deal with this problem, we propose the FN-VNS algorithm, inspired by the metaheuristic VNS. It includes a novel high-quality initial solution generator and an integrated perturbation strategy that provides diversity to four different local searches. These local searches are performed by means of modern parallel GPUs, which provide today the lower cost for high-performance processing units.

We compared the results obtained by FN-VNS with the results of the best algorithm developed for the problem. For 68 instances proposed in the literature, containing from 16 to 101 customers, FN-VNS was able to improve the results of 51 instances and be equal in 7 instances. In terms of efficiency, our parallel version of FN-VNS outperformed the sequential version for all tested instances, except one. The speedups compared to the sequential FN-VNS varied up to 14.49 times, achieving better results as the size of the instances grows. We also evaluated the influence of the parallelisation of the neighbourhoods in the execution time. In this evaluation, we used a new set of instances that we created to represent larger size problems available online, containing from 151 to 484 customers. For these instances, our parallel implementation obtained remarkable average speedups ranging from 45.74 up to 76.84 for the neighbourhood searches.

As future work, the proposed FN-VNS could be extended to solve related reverse logistics vehicle routing problems, for instance, including time windows in the problem and solving the multi-vehicle variant of the problem. Some changes in the infeasibility function can be studied in order to allow a more efficient calculation of the objective function values in both CPU and GPU. From the supply chain manager point of view, a real-time framework for route recomputation could be incorporated to the proposed algorithm. This would require not only a communication with the navigation system of the vehicles, but also a more sophisticated parallel solution using multiple GPUs to increase the acceleration of the developed algorithm.

Acknowledgements

The authors acknowledge the support from the Brazilian agencies CAPES, CNPq, FAPERJ and FAPEMIG. Thanks to the anonymous referees for their valuable contribution to this paper.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Brazilian agencies CAPES; CNPq; FAPERJ and FAPEMIG.

Note

1. Instances are available online at the author's website.

References

Allenby, Braden. 1993. "Supporting Environmental Quality: Developing an Infrastructure for Design." *Environmental Quality Management* 2 (3): 303–308.

Balas, Egon. 2004. "The Prize Collecting Traveling Salesman Problem and its Applications." Chap. 14 in *The Traveling Salesman Problem and its Variations*. Vol. 12, Combinatorial Optimization, edited by D. Z. Du, P. M. Pardalos, G. Gutin, and A. Punnen, 663–695. Boston: Springer. doi:10.1007/0-306-48213-4 14.

Belenguer, José-Manuel, Enrique Benavent, Christian Prins, Caroline Prodhon, and Roberto Wolfler Calvo. 2011. "A Branch-and-cut Method for the Capacitated Location-routing Problem." *Computers & Operations Research* 38 (6): 931–941.

Bettac, Evelyn, Karen Maas, Patrick Beullens, and Reinhold Bopp. 1999. "RELOOP: Reverse Logistics Chain Optimisation in a Multi-user Trading Environment." In *Proceedings of the 1999 IEEE International Symposium on Electronics and the Environment. ISEE-1999*, 42–47. Danvers, MA.

Beullens, Patrick, Dirk Van Oudheusden , and Luk N. Van Wassenhove. 2004. "Collection and Vehicle Routing Issues in Reverse Logistics." In *Reverse Logistics*, edited by R. Dekker, M. Fleischmann, K. Inderfurth, and L. N. Van Wassenhove, 95–134. Berlin: Springer.

Bras, Bert, and Mark W. McIntosh. 1999. "Product, Process, and Organizational Design for Remanufacture – An Overview of Research." *Robotics and Computer-Integrated Manufacturing* 15 (3): 167–178.

Bruck, Bruno P., André G. dos Santos, and José E. C. Arroyo. 2012. "Hybrid Metaheuristic for the Single Vehicle Routing Problem with Deliveries and Selective Pickups." In *Proceedings of the WCCI 2012 IEEE World Congress on Computational Intelligence*, 10–15. Brisbane, Australia.

Carter, Craig R., and Lisa M. Ellram. 1998. "Reverse Logistics – A Review of the Literature and Framework for Future Investigation." *Journal of Business Logistics* 19 (1): 85–102.

Coelho, Igor M., Luiz A. Pablo, Matheus N. Munhoz, Marcone Jamilson Haddad, F. Souza, and Luiz S. Ochi. 2012. "A Hybrid Heuristic Based on General Variable Neighborhood Search for the Single Vehicle Routing Problem with Deliveries and Selective Pickups." *Electronic Notes in Discrete Mathematics* 39: 99–106.

Coelho, Igor M., Luis S. Ochi, Pablo L. A. Munhoz, Marcone J. F. Souza, and Luiz S. Ochi. 2012. "The Single Vehicle Routing Problem with Deliveries and Selective Pickups in a CPU–GPU Heterogeneous Environment." In *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, June, 1606–1611. Liverpool: IEEE.

Dalé, Luíse Bispo, Lucas da Costa, Bonacina Roldan, and Peter Bent Hansen. 2013. "Analysis of sustainability Incorporation by Industrial Supply Chain in Rio Grande do Sul State (Brazil)." *Journal of Operations and Supply Chain Management* 4 (1): 25–36.

Dantzig, George B., and John H. Ramser. 1959. "The Truck Dispatching Problem." *Management Science* 6: 80–91.

De Brito, Marisa P., and Rommert Dekker. 2004. *A Framework for Reverse Logistics*. Berlin: Springer.

Dekker, Rommert, and Moritz Fleischmann. 2004. *Reverse Logistics: Quantitative Models for Closed-loop Supply Chains*. Berlin: Springer.

Derbel, Houda, Bassem Jarboui, Habib Chabchoub, Said Hanafi, and Nenad Mladenovic. 2011. "A Variable Neighborhood Search for the Capacitated Location-routing Problem." In *4th International Conference on Logistics (LOGISTIQUA-2011)*, 514–519. Hammamet.

Dethloff, Jan. 2001. "Vehicle Routing and Reverse Logistics: The Vehicle Routing Problem with Simultaneous Delivery and Pick-up." *OR-Spectrum* 23 (1): 79–96.

Farahani, Reza, Shabnam Rezapour, and Laleh Kardar, eds. 2011. *Logistics Operations and Management: Concepts and Models*, London: Elsevier.

Fleischmann, Mortiz, Hans Ronald Krikke, Rommert Dekker, and Simme Douwe P. Flapper. 2000. "A Characterisation of Logistics Networks for Product Recovery." *Omega* 28 (6): 653–666.

Ghodsi, Reza , and Alireza Shamekhi Amiri. 2010. "A Variable Neighborhood Search Algorithm for Continuous Location Routing Problem with Pickup and Delivery." In *Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation (AMS-2010)*, 199–203.

Golden, Bruce L., Subramanian Raghavan, and Edward A. Wasil, eds. 2008. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Vol. 43, Operations Research/Computer Science Interfaces Series. New York: Springer.

Green, Kenneth W., Pamela J. Zelbst, Jeramy Meacham, and Vikram S. Bhadauria. 2012. "Green Supply Chain Management Practices: Impact on Performance." *Supply Chain Management: An International Journal* 17 (3): 290–305.

Gribkovskaia, Irina, Gilbert Laporte, and Aliaksandr Shyshou. 2008. "The Single Vehicle Routing Problem with Deliveries and Selective Pickups." *Computers & Operations Research* 35 (9): 2908–2924.

Gungor, Askiner, and Surendra M. Gupta. 1999. "Issues in Environmentally Conscious Manufacturing and Product Recovery: A Survey." *Computers & Industrial Engineering* 36 (4): 811–853.

Gutiérrez-Jarpa, Gabriel, Vladimir Marianov, and Carlos Obreque. 2009. "A Single Vehicle Routing Problem with Fixed Delivery and Optional Collections." *IIE Transactions* 41 (12): 1067–1079.

Hansen, P., N. Mladenović, and J. M. Pérez. 2010. "Variable Neighbourhood Search: Methods and Applications." *Annals of Operations Research* 175: 367–407. doi:10.1007/s10479-009-0657-6.

Hansen, Pierre, Bo Hegedahl, Soren Hjortkjaer, and Børge Obel. 1994. "A Heuristic Solution to the Warehouse Location-routing Problem." *European Journal of Operational Research* 76 (1): 111–127.

Hassan, Mohsen M. D., Gary L. Hogg, and Donald R. Smith. 1986. "SHAPE: A Construction Algorithm for Area Placement Evaluation." *International Journal of Production Research* 24 (5): 1283–1295.

Janiak, A., W. Janiak, and M. Lichtenstein. 2008. "Tabu Search on GPU." *Journal of Universal Computer Science* 14 (14): 2416–2427.

Koopmans, Tjalling C., and Martin Beckmann. 1957. "Assignment Problems and the Location of Economic Activities." *Econometrica: Journal of the Econometric Society*:53–76.

Kumar, Sameer, Steve Teichman, and Tobias Timpernagel. 2012. "A Green Supply Chain is a Requirement for Profitability." *International Journal of Production Research* 50 (5): 1278–1296.

Laporte, Gilbert, and Yves Nobert. 1981. "An Exact Algorithm for Minimizing Routing and Operating Costs in Depot Location." *European Journal of Operational Research* 6 (2): 224–226.

Lin, S., and B. W. Kernighan. 1973. "An Effective Heuristic Algorithm for the Traveling-salesman Problem." *Operations Research* 21 (2): 498–516. doi:10.2307/169020.

Luong, T. V., L. Loukil, N. Melab, and E. Talbi. 2010. "A GPU-based Iterated Tabu Search for Solving the Quadratic 3-dimensional Assignment Problem. ACS/IEEE Intern." *Conference on Computer Systems and Applications*, 1–8.

Luong, The Van, Nouredine Melab, and El-Ghazali Talbi. 2013. "GPU Computing for Parallel Local Search Metaheuristic Algorithms." *IEEE Transactions on Computers* 62 (1): 173–185.

Martello, Silvano, and Paolo Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. New York: John Wiley & Sons.

Melechovskỳ, Jan, Christian Prins, and Roberto Wolfler Calvo. 2005. "A Metaheuristic to Solve a Location-routing Problem with Non-linear Costs." *Journal of Heuristics* 11 (5–6): 375–391.

Miller, Clair E., Richard A. Zemlin, and Albert W. Tucker. 1960. "Integer Programming Formulation of Traveling Salesman Problems." *Journal of the ACM (JACM)* 7 (4): 326–329.

Mladenović, Nenad, and Pierre Hansen. 1997. "Variable Neighborhood Search." *Computers & Operations Research* 24 (11): 1097–1100.

Nagy, Gábor, and Saïd Salhi. 2007. "Location-routing: Issues, Models and Methods." *European Journal of Operational Research* 177 (2): 649–672.

Nguyen, Thi Van Ha. 2012. "Development of Reverse Logistics - Adaptability and Transferability." PhD thesis, Technische Universitat Darmstadt.

Or, Ilhan. 1976. "Traveling Salesman-type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking." PhD thesis, Xerox University Microfilms.

Osman, Ibrahim Hassan. 1993. "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem." *Annals of Operations Research* 41 (1–4): 421–451.

Privé, Julie, Jacques Renaud, Fayez Boctor, and Gilbert Laporte. 2005. "Solving a Vehicle-routing Problem Arising in Soft-drink Distribution." *Journal of the Operational Research Society* 57 (9): 1045–1052.

Prodhon, Caroline, and Christian Prins. 2014. "A Survey of Recent Research on Location-routing Problems." *European Journal of Operational Research* 238 (1): 1–17.

Psaraftis, Harilaos N. 1983. "k-Interchange Procedures for Local Search in a Precedence-constrained Routing Problem." *European Journal of Operational Research* 13 (4): 391–402. http://www.sciencedirect.com/science/article/pii/0377221783900991.

Rogers, Dale S., and Ronald Tibben-Lembke. 2001. "An Examination of Reverse Logistics Practices." *Journal of business logistics* 22 (2): 129–148.

Santos, L. F. M., D. Madeira, E. Clua, S. Martins, and A. Plastino. 2010. "A Parallel GRASP Resolution for a GPU Architecture." In *International Conference on Metaheuristics and Nature Inspired Computing, META10*. Djerba Island.

Sarkis, Joseph, and Abdul Rasheed. 1995. "Greening the Manufacturing Function." *Business Horizons* 38 (5): 17–27.

Schulz, Christian. 2013. "Efficient Local Search on the GPU-investigations on the Vehicle Routing Problem." *Journal of Parallel and Distributed Computing* 73 (1): 14–31.

Sifaleras, Angelo, Ioannis Konstantaras, and Nenad Mladenovi. 2015. "Variable Neighborhood Search for the Economic Lot Sizing Problem with Product Returns and Recovery." *International Journal of Production Economics* 160: 133–143. http://www.sciencedirect.com/science/article/pii/S092552731400317X.

Singh, Surya P., and Renduchintala R. K. Sharma. 2006. "A Review of Different Approaches to the Facility Layout Problems." *The International Journal of Advanced Manufacturing Technology* 30 (5–6): 425–433.

Srivastava, Samir K. 2007. "Green Supply-chain Management: A State-of-the-art Literature Review." *International journal of management reviews* 9 (1): 53–80.

Subramanian, A., E. Uchoa, and L. Ochi. 2013. "A Hybrid Algorithm for a Class of Vehicle Routing Problems." *Computers & Operations Research* 40 (10): 2519–2531.

Subramanian, Anand, Lúcia M. A. Drummond, Cristiana Bentes, Luis S. Ochi, and Ricardo Farias. 2010. "A Parallel Heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery." *Computers & Operations Research* 37 (11): 1899–1911.

Sural, Haldun, and James H. Bookbinder. 2003. "The Single-vehicle Routing Problem with Unrestricted Backhauls." *Networks* 41: 127–136.

Thierry, Martijn Christiaan. 1997. "An Analysis of the Impact of Product Recovery Management on Manufacturing Companies." PhD thesis, Rotterdam School of Management (RSM), Erasmus University.

Toktay, L. Beril, Lawrence M. Wein, and Stefanos A. Zenios. 2000. "Inventory Management of Remanufacturable Products." *Management Science* 46 (11): 1412–1426.

Tompkins, James A., and Ruddel Reed Jr. 1976. "An Applied Model for the Facilities Design Problem." *The International Journal Of Production Research* 14 (5): 583–595.

Toth, Paolo, and Daniele Vigo, eds. 2001. *The Vehicle Routing Problem*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Validi, Sahar, Arijit Bhattacharya, and P. J. Byrne. 2014a. "A Case Analysis of a Sustainable Food Supply Chain Distribution System – A Multi-objective Approach." *International Journal of Production Economics* 152: 71–87.

Validi, Sahar, Arijit Bhattacharya, and P. J. Byrne. 2014b. "Integrated Low-carbon Distribution System for the Demand Side of a Product Distribution Supply Chain: A Doe-guided Mopso Optimiser-based Solution Approach." *International Journal of Production Research* 52 (10): 3074–3096.

Hoek, Van, and I. Remko. 1999. "From Reversed Logistics to Green Supply Chains." *Supply Chain Management: An International Journal* 4 (3): 129–135.

Vidal, Thibaut, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. 2013. "Heuristics for Multi-attribute Vehicle Routing Problems: A Survey and Synthesis." *European Journal of Operational Research* 231 (1): 1–21. http://www.sciencedirect.com/science/article/pii/S0377221713002026.

Winsemius, Pieter, and Ulrich Guntram. 1992. "Responding to the Environmental Challenge." *Business Horizons* 35 (2): 12–20.

Wong, M, and T Wong. 2009. "Implementation of Parallel Genetic Algorithms on Graphics Processing Units." In *Intelligent and Evolutionary Systems*, edited by M. Gen, D. Green, O. Katai, B. McKay, A. Namatame, R.A. Sarker, B.-T. Zhang, 197–216. Berlin: Springer.

Yu, Vincent F., Shih-Wei Lin, Wenyih Lee, and Ching-Jung Ting. 2010. "A Simulated Annealing Heuristic for the Capacitated Location Routing Problem." *Computers & Industrial Engineering* 58 (2): 288–299.

Zhu, Qinghua, and Joseph Sarkis. 2007. "The Moderating Effects of Institutional Pressures on Emergent Green Supply Chain Practices and Performance." *International Journal of Production Research* 45 (18–19): 4333–4355.