Gazi Mühendislik Bilimleri Dergisi
2021, 7(2): 90-98
Konferans Bildirisi/Conference Paper
https://dergipark.org.tr/gmbd

A Novel Permutation Based Solution Representation Technique for Vehicle Routing Problems on GPUs

Erdener ÖZÇETİN*[a], Gürkan ÖZTÜRK [b]

[a,*] *Hitit Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği Bölümü, Çorum, Türkiye*
[b] *Eskişehir Teknik Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği Bölümü, Eskişehir, Türkiye*

ARTICLE INFO

Recevied: 10.01.2021
Accepted: 03.08.2021

Keywords:
Vehicle routing problem
parallel programming
GPU
CUDA

ABSTRACT

In this study, the vehicle routing problem (VRP) which is a well-known NP-hard combinatorial optimization problem is handled on graphic processing units (GPUs). Solving optimally any kind of VRP is extremely hard when the instance size is large. For this reason, researchers tend to solve the VRP with meta-heuristics. Although, many well-designed meta-heuristics produce near-optimal solutions in reasonable time, still a challenge to solve large scale instances. To accomplish this issue, researchers need novel, fast and wisely designed parallel operators for the proposed algorithms. Furthermore, the success of these operators directly depends on the way the solution is represented. This paper offers a new permutation based solution representation technique (π+) for vehicle routing problems on GPUs. Results show that proposed technique can be used in many algorithms to accelerate computations.

https://dx.doi.org/10.30855/gmbd.2021.02.02

Araç Rotalama Problemleri için Grafik İşlem Birimleri Üzerinde Yeni Bir Çözüm Gösterim Tekniği

MAKALE BİLGİSİ

Alınma: 10.01.2021
Kabul: 03.08.2021

Anahtar Kelimeler:
Araç rotalama problemi
Paralel programlama
GPU
CUDA

ÖZ

Bu çalışmada, NP-Hard kombinatorik optimizasyon problemlerinden olan araç rotalama problemi (ARP), grafik işlem birimleri (GPU) üzerinde ele alınmıştır. Problem boyutunun büyümesiyle birlikte ARP'nin herhangi bir türünü optimal olarak çözmek oldukça zorlaşmaktadır. Araştırmacılar bu yüzden metasezgisel yöntemlere yönelmektedir. Her ne kadar bu metasezgisel algoritmalar kabul edilebilir sürelerde optimale yakın sonuçlar üretse de büyük boyutlu problemler için bu durum farklıdır. Bu durumu aşmak için, araştırmacılar önerilen algoritmalar için yeni, hızlı ve akıllıca tasarlanmış paralel operatörlere ihtiyacı bulunmaktadır. Bu operatörlerin başarısı doğrudan çözümün temsil edilme şekline bağlıdır. Bu makale, ARP'yi GPU'lar üzerinde etkin bir şekilde ele alabilmek için yeni bir permütasyon tabanlı çözüm gösterim tekniği (π +) sunmaktadır. Sonuçlar, önerilen tekniğin hesaplamaları hızlandırmak için birçok algoritmada kullanılabileceğini göstermektedir.

https://dx.doi.org/10.30855/gmbd.2021.02.02

1. INTRODUCTION *(GİRİŞ)*

The vehicle routing problem (VRP) can be defined as the finding optimal routes of distribution plans with side constraints [1]. Since the VRP introduced by Dantzig and Ramser [2], researchers have studied many variants of problem. This can be inferred that popularity of the VRP still ongoing since many

*__Corresponding author__: erdenerozcetin@hitit.edu.tr

To cite this article: Özçetin, E., Öztürk, G. "A Novel Permutation Based Solution Representation Technique for Vehicle Routing Problems on GPUs", *Gazi Journal of Engineering Sciences,* vol.7, no.2, pp. 90-98, 2021. doi:https://dx.doi.org/10.30855/gmbd.2021.02.02

applications of problem can be seen in real-life. Besides popularity, it is still a challenge to solve VRP optimally or near-optimally when the instance size increases.

Metaheuristic algorithms are generally cited as tools for solving certain types of optimization problems, such as combinatorial optimization problems like VRP. These algorithms do not guarantee the exact solution developed for optimization problems; but they are techniques that give satisfactory solutions in a short time. They are often used in cases where exact solution algorithms do not provide solutions in a reasonable time.

Whether it is exact method or meta-heuristic still requires intelligent and fast computational techniques. Especially, when the researchers study with very large scale problems, time is one of the most important issues to generate a good solution.

A graphics processing unit (GPU) is a processor designed to rapidly manipulate and alter memory to accelerate the visualization of images. As well as visualization, GPUs are increasingly used in computing. With the motivation of shortening the solution times of optimization problems, it is becoming widespread to develop metaheuristic algorithms on GPUs, mostly using by CUDA environment. However, this technology is an obstacle to the direct use of traditional programming knowledge and skills. In order to develop applications using this technology, it is necessary to understand the architecture of GPU very well.

The problem under consideration must be suitable for processing on the GPU. Furthermore, the representation of the problem and constraint handling will directly affect the efficiency of the GPU. In order to develop the algorithms on GPUs effectively, it is important to define proper representation to the problem structure.

In this study, we focus on solution representation of the VRP. We propose a new permutation based solution representation technique which we called π+. The main contributions of π+ can be summarized as follows:

- It is easy to manipulate.
- It offers a tremendous speed up opportunity on GPUs
- It can be adaptable for other combinatorial optimization problems.

The remainder of this paper is organized as follows. In Section 2, we review the VRP and meta-heuristic algorithms on GPUs. In Section 3, we describe the proposed permutation based solution representation technique and in Section 4 we show the computational experiments. In conclusion, we discuss the efficiency of the proposed technique and the potential for future studies.

2. VRP AND META_HEURISTICS on GPUS
(ARP ve GPU İÇİN METASEZGİSELLER)

Researchers commonly propose meta-heuristics algorithms for any variant of VRP. These algorithms can be single solution based such as tabu search, simulated annealing and variable neighborhood search or population based such as genetic algorithm and ant colony optimization. In many studies, these techniques can be hybridized and enriched with local search strategies to give both intensification and diversification properties to the algorithm.

2.1. VRP *(Araç Rotalama Problemi)*

There are many variants of the VRP in the literature such as capacity constrained, time windowed and open. When the VRP is mentioned, the first thing that comes to mind is the capacity constrained VRP. A vehicle fleet with the same capacity in the capacity-limited VRP moves from a warehouse and serves customers whose demands are known in advance. Each customer gets service from only one vehicle and each customer's demand is met. The vehicles cannot exceed their capacity and return to the warehouse at the end of service. It is aimed to find the least costly routes under these constraints. In one variant of the problem, there is the additional constraint that the travel time of each vehicle should not exceed a given travel time limit. In open VRP, each route starts at the depot and ends at a customer, visiting number of customers, each once, without returning to the depot.

Mathematical model of heterogeneous fleet open vehicle routing problem is given below.

Parameters:
$I, J = \{1, 2, \dots, n\}$: *set of customers*
$K = \{1,2, \dots, m\}$: *set of vehicles*
r_k : *capacity of kth vehicle*
d_j : *demand of jth customer*
c_{ijk} : *cost between jth and jth customer with kth vehicle*

Decision variables:

$$x_{ijk} = \begin{cases} 1, & \text{if } k^{th} \text{ vehicle } serve\ j^{th}\ cust.after\ i^{th}\ cust. \\ 0, & otherwise \end{cases}$$

$$y_{jk} = \begin{cases} 1, & \text{if } k^{th} \text{ vehicle } serve\ j^{th}\ customer \\ 0, & otherwise \end{cases}$$

Objective:

$$\min z = \sum_i \sum_j \sum_k c_{ijk} x_{ijk} \quad (1)$$

Constraints:

$$\sum_{i,i\neq j} x_{ijk} = y_{jk}\ ,\ \forall j,k\ ,j > 1 \quad (2)$$

$$\sum_{j,i\neq j,j>1} x_{ijk} \leq y_{ik}\ ,\ \forall\ i,k\ \ i > 1 \quad (3)$$

$$\sum_j \sum_k x_{1jk} \leq m \quad (4)$$

$$\sum_{j,j>1} d_j y_{jk} \leq r_k, \forall k \quad (5)$$

$$\sum_i \sum_k x_{ijk} = 1\ \forall j > 1 \quad (6)$$

$$u_{ik} - u_{jk} + r_k x_{ijk} \leq r_k - d_j, i \neq j, i > 1\ \&\ j > 1 \text{ ve } d_j + d_i \leq r_k \quad (7)$$

With the objective function (Eq. 1) the total cost is minimized. Constraint (Eq. 2) guarantees that jth customer will be served by kth, if $y_{jk} = 1$. Constraint set (Eq. 3) states that, if customer ith is visited by kth vehicle; then this vehicle may travel or may not travel to another customer. Constraint (Eq. 4) limits the number of vehicles. It is ensured not exceeding the capacities of the vehicles with (Eq. 5). Constraint set (Eq. 6) indicates that every customer must be visited exactly once. Relations (Eq. 7) are the sub-tour elimination constraints.

VRP diversifies according to the needs in application and gains new variants every day. Both technological innovations and the needs of the industry have led to the emergence of many different types of VRP. However, the need to make effective decisions in a short time has led researchers to various studies on VRP's solution methods.

The diversity of models and solution approaches in vehicle routing is enormous (see, e.g., [3] and [4]). For detailed information for variants of problem, readers are encouraged to see [5]. Some meta-heuristic approaches are commonly studied to solve the VRP like tabu search [6], variable neighborhood search [7] and ant colony optimization [8]. The number of metaheuristic algorithms in the literature can be easily increased, but in this study, we focus on the solution representations used in metaheuristic algorithm design for the VRP.

2.2. Meta-heuristics on GPUs *(Grafik İşlem Birimleri Üzerinde Metasezgisel Algoritmalar)*

Although metaheuristic algorithms search for solution spaces with highly advanced techniques, the increase in the size of the instances can cause algorithms to not converge to the best solution. Therefore, parallelization is a very important tool that allows the development of new methods for solving large problems.

Parallel computing has been an advanced way of solving difficult large-scale problems for years. The design and coding of sophisticated parallel metaheuristic algorithms are directly affected by the platform where the calculation will be made. GPU, which is used as a visualization tool in computer systems, is one of these platforms.

It is a fact that metaheuristic algorithms significantly reduce the computational complexity and perform the search for the best solution, but the increase in the size of the problems causes the solution times to increase and the solution quality to decrease. In combinatorial optimization problems, the increase in the instance size causes the solution space to grow exponentially. However, computing the objective function and many other calculations require more processing time. For these reasons, a parallel design to metaheuristic algorithms comes to the fore. In a broader sense, parallel and distributed computing on metaheuristic algorithms aims to:

- Speeding up the search,
- Improving the quality of the solution obtained,
- Increasing consistency,
- Solving large scale problems.

With these purposes, parallelization in metaheuristic algorithms occurs in three main dimensions:

- Solution level parallelization: This type of parallelization involves the simultaneous computation of the evaluation of a single solution by breaking it into pieces.
- Parallelization at the level of iteration: It means that multiple solutions can be calculated simultaneously or that the neighborhood comparison on a solution can be partitioned and evaluated simultaneously.
- Algorithmic level parallelization: It is the simultaneous processing of different metaheuristic algorithms in order to obtain more consistent results.

2.2.1. Literature review *(Literatür incelemesi)*

When we focus on combinatorial optimization problems that studied on GPUs, there are several publications in the literature. Tsutsui and Fujimoto [9] presented a parallel genetic algorithm on GPUs for the quadratic assignment problem (QAP) in their study. In this study, 3 to 12 times faster results were obtained than the serial algorithm. In another study [10], the same authors presented a method for the solution of the quadratic assignment problem, consisting of optimizing ant colonies and combining the 2-opt local search algorithm. According to the results, it was emphasized that the solution times on the GPUs are 24.6 times faster than the serial one. Czapinski [11] developed multiple onset tabu search algorithm for the QAP in the CUDA environment in their paper. In this study, some of the literature problems developed for the QAP were addressed and results were obtained in up to 70 times shorter time with solutions that averagely have 1% gaps to optimal solutions. The same problems also studied on GPUs in [12] and [13].

Cecilia et al. [14] developed an ant colonies optimization for the travelling salesman problem and obtained solutions up to 20 times faster. In another study conducted for the same problem, Delevacq et al. [15] achieved an acceleration up to 23.6 by ant colonies optimization.

Although the literature is rich on GPU implementations of population-based metaheuristics, only a few publications discuss routing problems. Shulz [16] presented a study on applications of local search algorithms for VRP. In the study, the memory operations were emphasized which is one of the bottlenecks when operating on GPUs. In another study for the VRP, Groer et al. [17] have applied local search techniques on GPUs. In another study, Szymon and Dominik [18] developed a parallel tabu search algorithm on GPUs for the multi-criteria VRP.

Local Search is a basic algorithm in discrete optimization and used commonly with metaheuristics. With the availability of CUDA, the number of papers studying local search based metaheuristics on the GPU increased. The technical report by Luong et al. [19] discusses a CUDA based GPU implementation of local search. The neighborhood evaluation is the most computationally intensive task in local search based algorithms such as variable neighborhood search. In many applications, selecting the best move is not always done on the GPU to use memory operation efficiently. The papers (Luong et al. [19], O'Neil et al. [20], Coelho et al. [21], Rocki and Suda [22]) all discuss different implementation details and CUDA specific optimizations.

The strategy that used for handling of constraints and objective components for each solution in the neighborhood is an almost perfect parallel task, see for instance [23] and [24] for an illustrating example.

2.2.2. Handling VRP with meta-heuristics *(ARP'nin metasezgisellerle ele alınması)*

There are some solutions representation methods for VRP, such as permutation based and binary encoding. An example of permutation based representation can be seen for VRP in Figure 1. It is known that representation methods for VRP strictly depends on the constraints of problem. In addition to this, time consuming operations can be handled according to representation of problem.

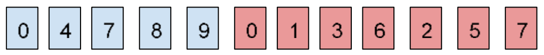


Figure 1: An example of permutation representation for VRP *(ARP için permütasyon tabanlı gösterime bir örnek)*

When it comes to parallel algorithms and concurrent computing, representation design of the problem comes to the fore. Davidovic et al. [25] offers the permutation based representation the best for concurrent computation on many types of problems.

3. Proposed Solution Approach *(Önerilen Çözüm Yaklaşımı)*

In many CPU-GPU oriented meta-heuristic algorithms for combinatorial optimization problems like the VRP are designed as follows: The starting solutions are generated and some complex operations are made on the host (CPU side). Then the solution is manipulated and sent to device (GPU side) to make parallel computations. This procedure brings many memory operations between host and device which makes GPUs inefficient.

Presentation of solutions on the GPU is extremely critical in terms of the simultaneous operation of the algorithm. π^+ has been proposed to show the solutions on the GPU, which provides many advantages.

3.1. Representation *(Gösterim)*

The main purpose of the π^+ is to standardize route lengths. We used dummy demand points ($\emptyset$) to ensure that all routes have equal lengths. The distance of all demand points including the warehouse to $\emptyset$ is *0*, and the distance of *Ø* to all demand points is a very large number (*M*). At the same time, the distance of *Ø* to itself is *0*. However, *Ø* does not have demand.

To make the new representation format easy to understand, we will continue to explain π$^+$ on open VRP. As it is mentioned before, in open VRP, a route starts from depot and ends at a customer. In this example, we assume that all the vehicles are identical. Namely, they have the same capacities. Except for the warehouse and dummy demand points, the demands are arranged in ascending order to obtain the maximum number of different demand points that can be served with one vehicle. The number *l* is obtained by adding *2* to this number. The reason for adding *2* to this value is that, in accordance with open VRP, the first elements in each route are created from the warehouse last elements and the dummy demand points.

0	3	1	4	5	Ø	0	6	2	7	Ø	Ø

Figure 2: An example of π$^+$ for open VRP *(AARP için bir π$^+$ örneği)*

The total length of a π$^+$ is obtained by multiplying the *l* by the number of vehicles. An example of π$^+$ with 2 vehicles and 8 nodes including the warehouse is as in Figure 2. Accordingly, it is assumed that a maximum of 4 different demands can be loaded on a vehicle according to the vehicle capacity by listing the demands ascending. Therefore, the number *l* is determined as 6. In π$^+$ every *l* segment represents a route.

For this example, the objective function of π$^+$ can be calculated as $\sum_{i=0}^{n-1} cost[\pi^+[i], \pi^+[i+1]]$. Please note that only a dummy demand point is added as the last column and last row in the cost or distance matrix.

3.2. Objective Function Calculation *(Amaç fonksiyonu hesaplaması)*

Objective function calculation is a critical issue for meta-heuristics. Depending on the flow of the algorithm and type of the problem, the objective function may need to be calculated over and over again. π$^+$ can reduce computation time for many combinatorial optimization problems like the open VRP on GPUs. This should be considered that each route designed as a block on grid structure of GPU. A vector can be defined on device for π$^+$ and each element of this vector will keep a route's cost.

Two random instances were created to show the effectiveness of the proposed solution representation method in calculating the parallel objective function. According to this, the instance A has 79 customers and a depot with 10 vehicles, and the instance B has 385 customers and a depot with 47 vehicles. We generate random solutions between 2^{10} and 2^{20}. Time of generating these solutions were discarded from comparison. On the other hand, we took into account the time of data transfers to GPU side for fair comparison. The comparisons for these random instances are shown in seconds in Table 1. This can be said that the objective function evaluation on GPUs with π$^+$ for open VRP offers a tremendous speed-up. In addition, it should not be overlooked that algorithms do not only consist of calculating the objective function, but also have many other computational operators. For this reason, we focused also other operators like local search.

Table 1: Performance comparison on objective function calculation *(Amaç fonksiyonu bakımından performans karşılaştırması)*

	A		B	
Number of solutions	n80k10 CPU (secs)	n80k10 GPU (secs)	n386k47 CPU (secs)	n386k47 GPU (secs)
2^{10}	0.126	0.085	0.356	0.097
2^{11}	0.154	0.088	0.620	0.100
2^{12}	0.300	0.089	1.412	0.104
2^{13}	0.529	0.094	2.500	0.106
2^{14}	1.162	0.096	5.528	0.143
2^{15}	2.285	0.136	10.027	0.152
2^{16}	4.383	0.142	20.497	0.210
2^{17}	8.887	0.151	42.560	0.351
2^{18}	20.770	0.225	84.241	0.676
2^{19}	34.871	0.365	174.168	1.378
2^{20}	73.872	0.649	372.455	3.282

3.3. Local Search Operators *(Yerel arama operatörleri)*

Most of well-designed meta-heuristics use local search operators. We can divide local search algorithms for VRP into intra-route and inter-route operators. Intra-route algorithms search for some neighborhoods in a route and inter-route algorithms search for neighborhoods between two or more routes.

3.3.1. Intra-route local search operators *(Rota içi yerel arama operatörleri)*

The most popular intra-route local search algorithms are Or-opt, 2-opt or 3-opt for the VRP. We implemented 2-opt and Or-opt for open VRP on GPUs to show the efficiency of π$^+$.

Since a similar strategy is followed in all neighborhoods, the operation on GPU is explained in detail only over the 2-opt neighborhood. Breaking *(i, j)* and *(i + 1, j + 1)* connections by breaking *(i, i + 1)* and *(j, j + 1)* connections with $j \geq i + 2$ in each route, simultaneously on all routes is tested. The inverse conversion between *(i + 1, j + 1)* performing at the appropriate ones is also performed simultaneously (see Figure 3)

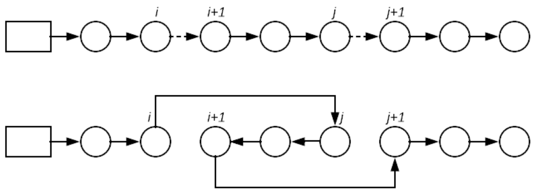


Figure 3. 2-opt neighborhood *(2-opt komşuluğu)*

In π^+, the warehouse that states at the beginning of the routes and the *Ø* that states at the end of routes are excluded from the local search. However, in π^+, *Ø* can also take place somewhere else the last point. When looking at the 2-opt neighborhood, the *Ø* in the route is prevented from getting ahead of a demand point, thanks to the large number of the distance of *Ø* to the demand points in the cost matrix. This block level parallelization can bring tremendous speed-up to intra-route local search operators.

3.3.2. Inter-route local search operators *(Rotalar arası yerel arama operatörleri)*

Basically, inter-route algorithms make moves between two or more vehicles. When concurrent search between vehicles is concerned, bank conflict should be taken into consideration. For GPUs the local memory is divided into memory banks. Each bank can only address one dataset at a time, so if a half-warp tries to load/store data from/to the same bank the access has to be serialized. In detail, this is not possible to compare a vehicle with more than one vehicle and apply meaningful changes in simultaneous operation (see Figure 4).

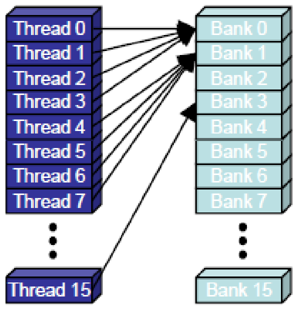


Figure 4. Bank conflict example *(Bank uyuşmazlığı örneği)*

For example, we suppose a one-to-one change (see Figure 5) inter-route search is performed on an example with three routes (R=Route, *R1-R2*, *R1-R3)*. In this case, in the comparisons of *R1-R2* and *R1-R3*, an element from *R1* may be replaced by an element from both *R2* and *R3*, and this causes the algorithm to malfunction or bank conflict in technological term.

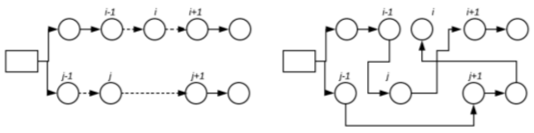
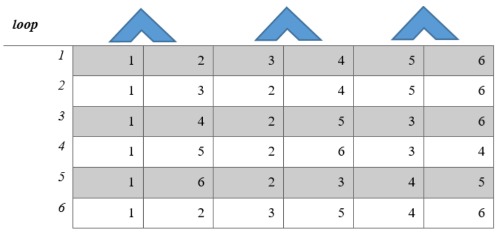


Figure 5. One-to-one change neighborhood *(Bire-bir değişim komşuluğu)*

In the proposed parallel method, comparisons up to *int ((number of vehicles) / 2)* during the search between vehicles are considered simultaneously. To compare all vehicles with each other a loop must be found in the code. In this case, there is a need for an index set that states which routes to compare in each repetition. Successively indexed routes are taken simultaneously for comparison. A reciprocal change of element between routes has been considered for each vehicle to be used at most once. A route cannot be processed more than once in a concurrent transaction. An example of the index set can be seen in Figure 6.



loop						
1	1	2	3	4	5	6
2	1	3	2	4	5	6
3	1	4	2	5	3	6
4	1	5	2	6	3	4
5	1	6	2	3	4	5
6	1	2	3	5	4	6

Figure 6. An example of index set *(İndeks kümesi örneği)*

4. Computational experiments *(Hesapsal deneyeler)*

With the proposed π^+ many calculations can be done in a parallel manner on GPUs. For a meta-heuristic algorithm that solves the VRP on graphic processing units can easily calculate objective function and run local search operators.

4.1. Test Environment *(Test ortamı)*

The platform for our investigations is a modern NVIDIA Maxwell architecture GPU (GTX 980) with 16 streaming processors, each with 128 cores, a total of 2048 cores for parallel computation. In addition, our workstation has an Intel Xeon E5-2630 2.4 Ghz. CPU and 32 GB RAM. Python 2.7 was used when coding the algorithms. Furthermore, PyCuda [26] library is used for GPU operations. Finally, kernels are coded as C ++.

4.2. Performance of local search operators *(Yerel arama operatörlerinin performansı)*

In this study, it is not aimed to design a total meta-heuristic algorithm for open VRP. However, we define a way to get performance of local search operators on GPUs by means of time. This should be considered that if we design an operator that has many conditional statements, the efficiency of GPUs would drop noticeably.

A wise performance evaluation method has been followed. We test intra-route (2-opt, Or-opt) and inter-route (one to one change) local search algorithms, respectively with different iterations. It is not allowed vehicle capacities to be exceeded, therefore we can say that we have a time-consuming process during the control of vehicle capacities in the exchange between routes. Furthermore, we consider the time of initial solution transfer to GPU side for fair comparison. Furthermore, in contrast to objective function evaluation we handle a single solution.

In Table 2, the results in seconds can be seen for different iterations for problem A and B. This can be said that memory transfer is a bottleneck for GPU side, up to 5000 iterations. This can be comprehended that when the number of iterations increases, things turn in favor of GPU. In the B problem with 386 customers and 47 vehicles, when the number of iterations is increased, an acceleration of up to five times is observed. As a consequence, a significant acceleration can be given to the algorithm with the proposed strategy.

Table 2. Performance comparison on local search operators *(Yerel arama operatörleri bakımından performans karşılaştırması)*

	A		B	
Iterations	n80k10 CPU (secs) (serial code)	n80k10 GPU (secs) (parallel code)	n386k47 CPU (secs) (serial code)	n386k47 GPU (secs) (parallel code)
1000	0.705	1.166	1.788	3.366
2500	1.655	1.754	3.670	4.785
5000	2.698	2.227	6.299	6.213
10000	5.993	3.566	14.365	9.137
25000	12.367	5.217	29.437	13.767
50000	22.335	7.425	56.432	18.998
100000	39.215	9.665	110.369	29.231

5. Conclusions and Further Research *(Sonuçlar ve gelecek çalışmalar)*

For high GPU utilization, there are a number of implementation issues, the most important ones being related to configuration and use of memory structures, and code diversion for configurable thread blocks. There is still a need to solve large combinatorial optimization problems like the VRP with fast meta-heuristic algorithms. GPU offers an opportunity to accelerate algorithms.

In this study, we propose a novel permutation-based solution representation technique which is easy to manipulate for the VRP. The proposed representation technique can be used with many meta-heuristic algorithms and many variants of the VRP. We showed the efficiency of this technique on objective function calculation and some local search operator. For the objective function calculation, the parallel implementation runs up to 100 times faster than the sequential one. A promising acceleration for the local search operators have also been obtained. For further studies, researchers can adapt the proposed representation technique to other combinatorial optimization problems such as quadratic assignment problem with new heuristic or meta-heuristic algorithms.

ACKNOWLEDGMENT *(TEŞEKKÜR)*

This work was funded with projects by Eskisehir Technical University, project numbers 1601F004 and 19ADP022.

CONFLICT OF INTEREST STATEMENT *(ÇIKAR ÇATIŞMASI BİLDİRİMİ)*

No potential conflict of interest was reported by the authors.

REFERENCES *(KAYNAKLAR)*

[1] Gilbert Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms", *European Journal of Operational Research*, vol. 5 no. 3, pp. 345-358, 1992. doi:https:// dx.doi.org/10.1016/0377-2217(92)90192-C

[2] G. Dantzig and J. Ramser, "The Truck Dispatching Problem", *Management Science*, vol. 6, pp. 80-91, 1959. doi: https://dx.doi.org/10.1287/mnsc.6.1.80

[3] P. Toth and D. Vigo, "The granular tabu search and its application to the vehicle-routing problem", *Informs Journal on Computing*, vol. 15, no. 4, pp. 333-346, 2003. doi: https://dx.doi.org/10.1287/ijoc.15.4.333.24890

[4] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems", *Computers & Operations Research*, vol. 34, pp. 2403-2435, 2007. doi: https://dx.doi.org/10.1016/j.cor.2005.09.012

[5] S. N. Kumar and R. Panneerselvam, "A Survey on the Vehicle Routing Problem and Its Variants", *Intelligent Information Management*, vol. 4, pp. 66-74, 2012. doi: https://dx.doi.org/10.4236/ iim.2012.43010

[6] E. Taillard, P. Badeau, M. Gendreau, F. Guertin and J.Y. Potvin, "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows", *Transportation Science*, vol 31, no. 2, pp. 101-195, 1997. doi:https://dx.doi.org/10.1287/trsc.31.2.170

[7] K. Fleszar, I. Osman and K. Hindi, "A variable neighbourhood search for the open vehicle routing problem", *European Journal of Operational Research*, vol. 195, pp. 803-809, 2009. doi: https://dx.doi.org/10.1016/j.ejor.2007.06.064

[8] Bin Yu, Zhong-Zhen Yang, Baozhen Yao, "An improved ant colony optimization for vehicle routing problem", *European Journal of Operational Research*, vol. 196, no. 1, pp. 171-176, 2009. doi:https://dx.doi.org/10.1016/j.ejor.2008.02.028

[9] Shigeyoshi Tsutsui and Noriyuki Fujimoto, "Solving quadratic assignment problems by genetic algorithms with GPU computation: a case study," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers (GECCO '09)*, New York, USA, 2009, pp. 2523–2530. doi:https://dx.doi.org/10.1145/1570256.1570355

[10] S. Tsutsui and N. Fujimoto, "Fast QAP solving by ACO with 2-opt local search on a GPU," *2011 IEEE Congress of Evolutionary Computation (CEC)*, 2011, pp. 812-819. doi: https://dx.doi.org/10.1109/CEC.2011.5949702.

[11] M. Czapinski, "An effective Parallel Multistart Tabu Search for Quadratic Assignment Problem CUDA platform", *J. Parallel Distrib. Comput.* , vol. 73, pp. 1461-1468, 2013. doi:https://dx.doi.org/10.1016/j.jpdc.2012.07.014

[12] E. Özçetin, G. Öztürk, "A Parallel Iterated Local Search Algorithm on GPUs for Quadratic Assignment Problem", *International Journal of Engineering Technologies*, vol. 4, no. 2, pp. 123-127, 2018.

[13] E. Özçetin, G. Öztürk, "A Hybrid Genetic Algorithm for the Quadratic Assignment Problem on Graphics Processing Units", *Anadolu University Journal of Science and Technology-A Applied Sciences and Engineering*, vol. 17, no. 1, pp. 167-180, 2016. doi: https://dx.doi.org/10.18038/btda.15399

[14] J. M. Cecilia, J. M. Garcia, A. Nisbet, M. Amos and M. Ujaldon, "Enhancing data parallelism for Ant Colony Optimization on GPUs", *J. Parallel Distrib. Comput.*, vol. 73, pp. 42-51, 2013. doi: https://dx.doi.org/10.1016/j.jpdc.2012.01.002

[15] A. Delevacq, P. Delisle, M. Gravel and M. Krajecki, "Parallel Ant Colony Optimization on Graphics Processing Units", *J. Parallel Distrib. Comput.*, vol. 73, pp. 52-61, 2013. doi: https://dx.doi.org/10.1016/j.jpdc.2012.01.003

[16] C. Shulz, "Efficient local search on the GPU-Investigations on the vehicle routing problem", *J. Parallel Distrib. Comput.*, vol. 73, no.1, pp. 14-31, 2013.

[17] C. Groer, B. Golden and E. Wasil, "A Parallel Algorithm for the Vehicle Routing Problem", *INFORMS Journal on Computing*, vol. 23, pp. 315-330, 2011. doi: https://dx.doi.org/10.1287/ ijoc.1100.0402

[18] J. Szymon and Z. Dominik, "Solving Multi-criteria Vehicle Routing Problem by Parallel Tabu Search on GPU", Procedia Computer Science, vol. 18, pp. 2529-2532, 2013.

[19] Thé Van Luong, Nouredine Melab, El-Ghazali Talbi. "GPU Computing for Parallel Local Search Metaheuristics", *IEEE Transactions on Computers,* vol. 62, no. 2, pp.173-185, 2013. doi: https://dx.doi.org/10.1109/TC.2011.206

[20] M.A. O'Neil, D. Tamir, M. Burtscher, "A parallel GPU version of the traveling salesman problem," in: *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, Lasvegas, Nevada, USA, 2011, pp.348-353.

[21] I. Coelho, L. Ochi, P. Munhoz, M.Souza, C. Bentes, R. Farias, "The Single Vehicle Routing Problem with Deliveries and Selective Pickups in a CPU-GPU Heterogeneous Environment", *International Journal of Production Research,* vol. 54, no. 4, 945-962, 2016. doi:https:// dx.doi.org/10.1080/00207543.2015.1035811

[22] K. Rocki and R. Suda, "Accelerating 2-opt and 3-opt Local Search Using GPU in the Travelling Salesman Problem" 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2012, pp. 705-706. doi: https://dx.doi.org/10.1109/CCGrid.2012.133.

[23] N. Melab, E.G. Talbi, S. Cahon, E. Alba, G. Luque, "Parallel Metaheuristics: Models and Frameworks," in *E.G. Talbi Parallel Combinatorial Optimization*, John Wiley & Sons, Inc., New York, 2006, pp. 149-161.

[24] André R. Brodtkorb, Trond R. Hagen, Christian Schulz, Geir Hasle, "GPU computing in discrete optimization. Part I: Introduction to the GPU", *EURO Journal on Transportation and Logistics*, vol. 2, no. 1–2, pp. 129-157, 2013. doi: https://dx.doi.org/10.1007/s13676-013-0025-1

[25] T. Davidović, P. Hansen, N. Mladenovic, "Permutation-based genetic, tabu and variable neighborhood search heuristics for multiprocessor scheduling with communication delays", *Asia Pacific Journal of Operational Research*, vol. 22, no.3, 2005. doi:https://dx.doi.org/10.1142/S021759590500056X

[26] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov ve A. Fasih, "PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation", Parallel Computing, vol. 38, no. 3, pp. 157-174, 2012.

NOT: This article is an expanded version of the paper presented at the International Conference on Engineering Technologies (ICENTE'20) has been organized in Konya, Turkey on 19-21 November 2020.