Journal of Natural Sciences and Mathematics
Qassim University, Vol. 13, PP 12-23 (January 2020)

Solving Capacitated Vehicle Routing Problem by Iterated Greedy (IG) algorithm

Abdulwahab Almutairi

School of Mathematics, University of Qassim

a.abdulwahab.a@gmail.com

§1.1 Abstract

One of most significant logistics problems in the field of transportation and distribution is the Capacitated Vehicle Routing Problem (CVRP). The VRP has received particular attention for many years. In general, the problem considers the vehicles routing with limited capacity from a central depot to a set of geographically dispersed customers, where real (actual) demand of each customer is independent and known in advance. To the best of our knowledge, there is no research reported in the literature that combines CWS and IG in order to solve the CVRP. Also, safety stock and reloading/restocking have been considered in order to minimize the total cost of the problem. The contribution of the paper is to compare the CWS with IG. Also, the methodology includes two stages. In the first stage, optimal a-priori routes are generated using CWS and then the methodology has been extended by applying IG algorithm. In addition, when a vehicle capacity is exceeded, recourse actions have to be planned/designed to ensure the feasibility of solutions in case of route failure. In conclusion, our computational experiments on benchmark instances show that the IG is able to generate better solutions than CWS.

Keywords: The Capacitated Vehicle Routing Problem (CVRP), Clarke and Wright Saving algorithm (CWS), and Iterated Greedy algorithm (IG).

§1.2 Introduction

Iterated Greedy (IG) algorithm considers one of the heuristics that has been implemented in the hybrid flow-shop scheduling side with the aim of minimising the make-span, and it is very simple to apply as shown by the experimental results. [10] has successfully developed several heuristics for solving the flow-shop scheduling problem such as IG algorithm. The IG algorithm includes two steps. In the first step, an initial solution is generated and in the second step, two phases will repeated named destruction phase, where several customers are chosen from the current solution and construction phase, and the chosen customers are inserted into the sequence, until the stopping criteria are met. In recent years, many algorithms have emerged to address problems such as scheduling and transportation and obtain better solutions. IG algorithm has been applied in the scheduling problems. Therefore, the aim of this paper is to apply IG algorithm in the transportation side in order to minimize the total route costs. However, when a vehicle capacity is exceeded, recourse actions (reloading/restocking) have to be planned/designed to ensure the feasibility of solutions in case of route failure. Transportation plays a significant role in our daily lives. Researchers have used different targets to measure and obtain optimal solutions. In recent decades, extensive research on VRP and associated scheduling problems has been carried out. Also, several solutions have been proposed in the literature for the optimal solution to VRP.

In general, VRP can be described as a problem that arises when designing either optimal collection or delivery routes from a central depot to a set of geographically dispersed customers where each customer has a non-negative demand that is served by a single vehicle. A number of identical or heterogeneous vehicles located in a central depot with identical capacity, can be operated by a number of drivers to distribute goods along the most appropriate road network. The main objective of the VRP is to minimize the total route cost. Here are the most traditional types of constraints that have to be satisfied:

- Each vehicle route starts at the central depot and returns at the same depot;
- Each customer is visited exactly once by one vehicle;
- The total demand of any vehicle route cannot exceed the maximum vehicle capacity.

§1.3 Literature review of IG

IG algorithm is one of the most significant heuristic methods that can deal with scheduling problems. Also, IG algorithm has been implemented to solve the flow-shop problem combined with other methods and the computational results have shown its performance. Papers [9] and [10] proposed an IG algorithm to solve different aspects of the flow-shop problem. For instance, IG algorithm can be proposed for solving a complex flow-shop problem by minimising the total weighted tardiness and minimising the maximum completion time. The fundamental concept behind using an IG algorithm is to iterate over constructive algorithms and the fundamental concept behind using Iterated Local Search (ILS) is that it resides in iterating over local searches in a particular way. Adding ILS into an IG algorithm can make the similarities between these algorithms become more pronounced. [9] proposed an IG algorithm to achieve a solution for the flow-shop problem with sequence dependent setup times, whereby the setup time depends on the job previously processed on each machine. Also, the ability of the proposed IG algorithm in finding feasible solutions within reasonable computational times, which are near to the optimum solutions, make it justifiable to use in the flow-shop problem with sequence dependent setup times. A simple and effective IG algorithm based on an NEH constructive heuristic has been implemented by [10] to solve the Permutation Flow Shop Scheduling Problem (PFSP). Also, both IG algorithm with and without ILS obtained good solutions. They provided a complete comparative evaluation of the effectiveness and efficiency of this method.

In addition, the IG algorithm is considered as a well-known algorithm and it provides very good results in a variety of applications. Many researchers proposed the IG algorithm to solve problems such as PFSP especially on flow-shop problem [10], so that the following sentences will present the IG algorithm process. The IG algorithm has basic steps: destruction, construction, optimisation, and acceptance criteria. The initial solution has to randomly generate a sequence of solutions by using the NEH algorithm, then the two main phases of destruction and construction will be implemented. Firstly, in the destruction phase, some solution components are selected and removed from a previous solution. The second step is the construction phase; after removing components from the destruction phase, the construction procedure is applied by re-assignment to different positions to achieve the minimum cost. This has to be done for each removed component. Local search can be added to improve each solution that is generated in the second step. The last step is the acceptance criteria, and the new solution is compared with the previous solution to choose the better solution with no constraint violation. [10] presented the pseudo code of IG algorithm with local search in order to solve the PFSP. Also, they used the random number distributed uniformly between [0, 1] to check the termination criterion, (the demon-like criterion). Thus, if the demon-like criterion is greater-than or equal-to the random number, then they replaced the current list of customers with the old one. They also showed the code implementation of an iterative improvement procedure, using the insertion neighbourhood algorithm, in order to improve the final solutions.

Some researchers have integrated an IG algorithm with other methods in order to solve the scheduling problems. For example, unrelated parallel machine scheduling was solved by using an IG algorithm based meta-heuristic that is able to find good quality solutions in a very short time [2]. The aim was to minimise the maximum completion time of the jobs. A variable neighbourhood descent algorithm is used to obtain an initial solution and it applied Insertion and Interchange local searches till local optimum is achieved. After that, IG algorithm is implemented to improve the solution. From the computational results discussion, they are able to select the jobs and machines in a smart way to achieve one of the good solutions. A comprehensive benchmark test of 1400 instances was tested in order to compare all proposed algorithms against state-of-the-art methodologies. Furthermore, computational results showed that these solutions are, most of the time, better than the current state-of-the-art methodologies, by a statistically significant margin.

[3] presented two variants of both ILS and IG algorithm to minimise the completion time in two machines PFSP. They have also taken a step further than previous researches providing a new priority rule based on the availability of two machines for the total completion time under time lag. Later, IG algorithm has been combined with simulated annealing by [1] to minimise the total completion time for the two machine flow-shop scheduling problem. Two of the advantages of using these two methods are that it is easy to deal with

during the implementation; it also provides a compromise between intensification and diversification, that improves the solution quality by accepting better solutions. They used simulated annealing to generate initial solutions then they applied IG to obtain the goal. The characteristic of this method is that it can accept the solutions that improve the current solution and also accept solutions that deteriorate the current solution. The idea behind IG is to generate a sequence of solutions and then choose the best solution.

The IG algorithm was proposed to solve different problems and it has shown to reach excellent performance on the permutation flow-shop problem with different criterion such as time dependence. Furthermore, it is able to improve some of the best known solutions on the permutation flow-shop problem. A large number of scheduling problems can benefit from ideas underlying IG algorithm. [5] proposed a modified version of IG algorithm to minimise the total cost, along with duration of convergence for task assignment problem. Also it capitalises on the efficacy of the Parallel Processing paradigm. The main target of the modification is summarised as follows: (1) to enhance the quality of assignment in every iteration, (2) to utilise the values from the preceding iterations and (3) at the same time assigning these smaller computations to internal processors to hasten the computation.

[6] proposed a new way: multi-objectives IG algorithm to solve track scheduling in cross-dock problems with temporary storage. For this problem a multi-objectives IG algorithm employs advanced features such as modified crowding selection, restart phase and local search. From the performance point of view, this proposed method showed better solutions and outperformed the other methods. Multi-objectives IG algorithm can deal with a population of non-dominated outcomes as a working set, instead of just a single outcome. An IG algorithm was developed by [8] to solve a job scheduling problem with zero buffer constraints and with two known variants: the block job shop scheduling with swap and without swap. A comparison with recent published results showed that an IG algorithm improved the best known solutions in a literature review on benchmark instances. From the computational results, it is also important to notice that the IG has a broad applicability since it can be easily applied to any complex scheduling problem modelled by means of the alternative graph formulation. Despite the massive research effort on the VRP and the success of using IG in some Combinatorial Optimisation Problems, there has not been any published paper on the use of IG algorithm to solve the VRP and its variants. For this reason, this algorithm has special attention in order to implement IG algorithm on VRP.

§1.4 Proposed Methodology

The IG algorithm was presented by [10] for the permutation flow shop scheduling problems. It has been applied successfully in different scheduling and other problems. According to [10], the IG algorithm is conceptually simple and efficient; it is based on generating a sequence of solutions by iterating over a greedy construction heuristic using two main phases, which are: destruction and construction. In the destruction phase, a number of solution elements are randomly selected to be removed from a previously constructed complete candidate solution. In the second phase, the construction procedure is applied, in which the elements that previously removed in the destruction phase are re-inserted into the set of remaining elements until all candidates are inserted. An acceptance criterion is applied once a candidate solution has been completed, to decide whether the new solution will replace the current solution. The process of IG is then iterated over these steps until a stopping criterion is met. A framework of the IG algorithm is given in Figure 1, where Clarke and Wright Saving algorithm (CWS) has generated the initial solution, and where d represents the randomly chosen number of the customers.

```
Procedure Iterated Greedy _for _VRPSD
$\pi := $ GenerateInitialSolution();
$\pi_{best} = \pi$
While (NotTermination)
        $\pi_1$=DestructionConstruction( $\pi$, d)
        if (f ($\pi_1$) < f($\pi$)) then
            $\pi_1 = \pi$
            if (f ($\pi$) < f ($\pi_{best}$)) then
                $\pi_{best} = \pi$
endwhile
return
endprocedure
```

Figure 1. Iterated Greedy Algorithm

In any IG algorithm, there are two main phases: destruction and construction. In the destruction phase, there are two sets $\pi^D$and π^R, where π^D represents the set of d customers that select randomly from the current solution, and π^R is the set of the remaining customers. The first sequence π^D with size d of customers is the original solution without the removed customers. The second sequence with the size $n - d$ of customers is denoted as π^R, including the removed customers in the order. The construction procedure begins with π^D and inserts the first customer of π^R into all the possible locations $n - d + 1$ of π^D. The best position for π^R in the augmented π^D sequence is the one that provides the smallest total cost C_{max}. The second customer is then considered and the process is repeated until π^R is empty. Figure 2 presents the procedure for destruction and construction in more detail. After finishing the destruction and construction phases, the acceptance criterion is used to consider whether the new sequence solution obtained is accepted or not as the current solution for the next iteration. One of the simplest acceptance criteria is to accept a new sequence only when the sequence achieves a better solution than the previous solution.

The IG approach presented above has been used to solve the VRP instances when all the information about each instance is known already. Many common restrictions that are either related to the vehicles or the customers are applied. For example, each vehicle has a limit capacity, each customer has a specific demand and the visits to each customer cannot be split twice. In order to meet the requirements of the route design, the objective function of minimizing the cost should be considered. This situation is termed as a deterministic case. This approach was also implemented in this study for a number of VRP instances from the literature.

```
Procedure Destruction_Construction (π, d)
Set $\pi^D$ empty,     % Destruction
for $i \leftarrow 1$ to $d$ do
        $\pi^D \leftarrow$ the set of $d$ selected customer randomly
endfor
        $\pi^R \leftarrow$ the remaining set of customers
for $j \leftarrow 1$ to $d$ do              % construction
        $\pi_{best}$= best cost obtained after inserting customer from $\pi^R$ in all possible
        positions of $\pi^D$
endfor
end
```

Figure 2. Destruction and construction

§1.5 Computational results

In this section, the methodology described in this paper was implemented as a Java application. In the experiments, a Macintosh HD on 2.3 GHz Intel HD Graphics 4000 1024 MB with Intel Core i7 with 4 GB 1600 MHz DDR3 was used to perform the computational experiments. The iterated greedy algorithm was tested in several instances originally developed for the Capacitated VRP (CVRP). A number of tests of well-known standard benchmarks described in the literature, for problems related to CVRP, were described, and the outputs obtained for total cost of this approach were compared between IG algorithm and randomised IG algorithm. A set of 55 classical CVRP instances are considered, namely instances "A", "B", "E", "F", and "P". The key characteristics of the test problems are described online at https://www.coinor.org/SYMPHONY/branchandcut/VRP/data/index.htm.old. For each instance, the location of the customer in terms of x and y coordinates and the number of the vehicles, were previously known. In addition, the central depot is located at (0,0) and the Euclidean distance was used. The numbers of both customers and vehicles can be inferred from the name of each instance in the benchmark problems. The vehicles' capacity has to be similar for all the available fleet serving the customer demand. For example, P-n101-k4 indicates that P represents the instance class and 101 represents the number of customers in this class, whilst 4 represents the number of the vehicles that will be used to serve these customers.

In this paper, several computational experiments designed to test the VRP instances were conducted. The vehicle maximum capacity has been considered as 100%. However, since no safety stock is used, there is a chance that solutions can suffer from route failures. Therefore, the recourse actions have been used in order to serve all customer demand. An example of recourse action is that the vehicle can return to the central depot when it is full to unload and then resume collections as planned. Another example is that the vehicle can return to the central depot when it is full, as in the previous example, and then re-optimise the remaining part of the planned route. Table 1 and 2 summarises the complete computational results of Clarke and Wright Saving algorithm and Iterated Greedy Algorithm for all 55 instances; the best results were found to use two different approaches for solving VRP with the same well-known benchmark problems.

The results obtained from the preliminary analysis of both approaches are presented in this table and the second column of the table shows the name of the instances used for all the algorithms. Column 3 is related to solutions obtained with the Clarke and Wright Saving algorithm (CWS) where the vehicle capacity is considered during the design stage as 100%. The fourth column is related to solutions obtained by Iterated Greedy algorithm (IG) approach. The gaps between these two approaches are presented in the last column. The last column shows the % improvement between the Clarke and Wright Saving algorithm (CWS) and Iterated Greedy (IG) algorithm. Comparative results between the approaches are shown in the last column whilst the bold numbers represent the instances that improved. The improvement rate is defined as: $Improvement\ rate\ =\ \frac{Final\ best\ solution\ -\ previous\ best\ solution}{Final\ best\ solution} * 100$. The bottom row in Table 2 shows the average value of the results for each approach for all instances of the problem.

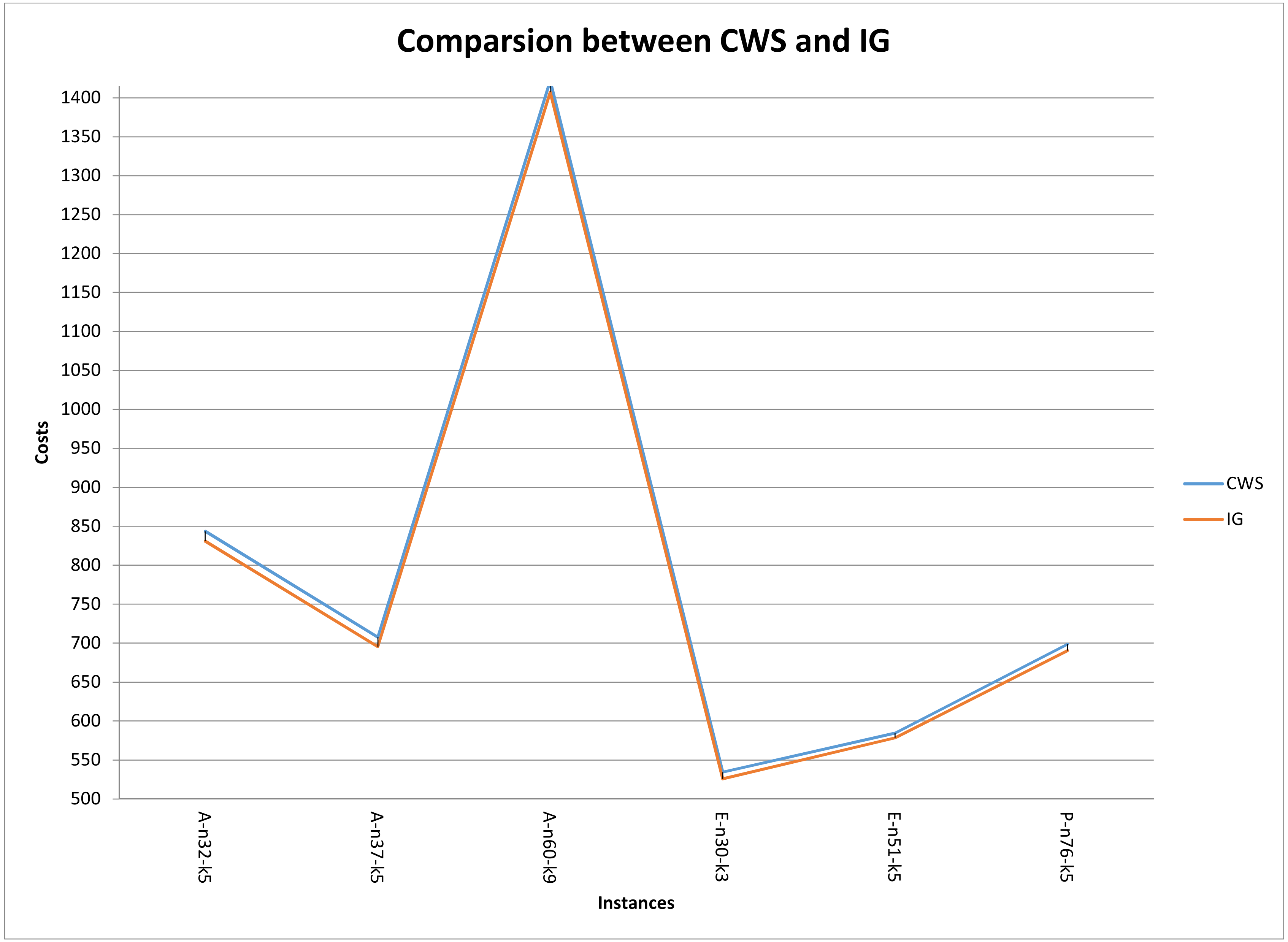
As shown in Tables 1 and 2, the most interesting column is the last column, which represents the improvement between the approaches. The improvement column shows that in most instances the improvement is generally small and does not reach 2% when these two approaches are compared – CWS with IG. For example, for instances that produced better results, A-n32-k5, A-n37-k5, A-n60-k9, E-n30-k3, E-n51-k5, E-n76-k7 and P-n76-k5 were found to be the best solutions and obtained results above 1%. From the tables, in terms of the improvement column, in 37 out of 55 instances the improvement of the total cost is less than 1% and in 6 out of 55 instances it is more than 1%. However, 12 out of 55 instances had no improvement. Notice that the smallest improvement, that is zero, has been omitted from the calculation of the average in the last row. In addition, the average value of the results between CWS and IG for all instances of the problem is equal to 0.53%.

Name of instance		CWS	IG	GAP (CWS – IG)
1	A-n32-k5	843.7	**830.7**	1.56%
2	A-n33-k5	712	**711.8**	0.02%
3	A-n33-k6	776.3	**776**	0.03%
4	A-n37-k5	707.8	**695.4**	1.78%
5	A-n38-k5	768.1	**765.9**	0.28%
6	A-n39-k6	863.1	**856.9**	0.72%
7	A-n45-k6	1006.5	1006.5	0.00%
8	A-n45-k7	1199.9	**1198.1**	0.15%
9	A-n55-k9	1099.8	**1098.5**	0.11%
10	A-n60-k9	1421.9	**1407.2**	1.04%
11	A-n61-k9	1102.2	**1094.5**	0.70%
12	A-n63-k9	1687.9	**1683.7**	0.24%
13	A-n65-k9	1239.4	**1239**	0.03%
14	A-n80-k10	1860.9	**1854.9**	0.32%
15	B-n31-k5	681.2	681.2	0.00%
16	B-n35-k5	978.3	**970.7**	0.78%
17	B-n39-k5	566.7	**565.4**	0.22%
18	B-n41-k6	898.1	**897**	0.12%
19	B-n45-k5	757.2	**755.2**	0.26%
20	B-n50-k7	748.8	**748.2**	0.08%
21	B-n52-k7	764.89	**762.5**	0.31%
22	B-n56-k7	733.7	**730.8**	0.39%
23	B-n57-k9	1653.4	**1651.9**	0.09%
24	B-n64-k9	921.6	**919.6**	0.21%
25	B-n67-k10	1099.9	**1098.9**	0.09%
26	B-n68-k9	1317.8	**1317.7**	0.01%
27	B-n78-k10	1264.6	**1260.6**	0.31%

Table 1. Comparison of methodologies for 55 selected VRP instances

Name of instance		CWS	IG	GAP (CWS – IG)
28	E-n22-k4	388.8	388.8	0.00%
29	E-n30-k3	534.4	**525.8**	1.63%
30	E-n33-k4	843.1	**842.8**	0.35%
31	E-n51-k5	584.6	**578.6**	1.03%
32	E-n76-k7	737.7	**724.2**	0.86%
33	E-n76-k10	900.3	**893.9**	0.71%
34	E-n76-k14	1073.4	**1072.9**	0.04%
35	F-n45-k4	745	**740**	0.67%
36	F-n72-k4	257	**255**	0.78%
37	F-n135-k7	1207	**1201**	0.49%
38	M-n101-k10	833.5	**825.3**	0.99%
39	M-n121-k7	1068.1	**1057.6**	0.99%
40	P-n19-k2	237.9	**236.5**	0.59%
41	P-n20-k2	233.9	233.9	0.00%
42	P-n22-k2	239.5	239.5	0.00%
43	P-n22-k8	590.6	590.6	0.00%
44	P-n40-k5	518.4	518.4	0.00%
45	P-n50-k8	674.3	674.3	0.00%
46	P-n50-k10	734.3	734.3	0.00%
47	P-n51-k10	790.9	790.9	0.00%
48	P-n55-k7	2441.7	2441.7	0.00%
49	P-n55-k15	978.1	978.1	0.00%
50	P-n60-k10	800.2	**796.2**	0.50%
51	P-n65-k10	851.7	**850.9**	0.09%
52	P-n70-k10	896.9	**894.8**	0.23%
53	P-n76-k4	689.1	**683.1**	0.87%
54	P-n76-k5	698.5	**690.4**	1.17%
55	P-n101-k4	765.4	**761.7**	0.48%
	Average	890.72	**887.27**	0.53%

Table 2. Comparison of methodologies for 55 selected VRP instances


Comparsion between CWS and IG
Costs
1400
1350
1300
1250
1200
1150
1100
1050
1000
950
900
850
800
750
700
650
600
550
500
A-n32-k5
A-n37-k5
A-n60-k9
E-n30-k3
E-n51-k5
P-n76-k5
Instances
CWS
IG

This chart presented the best solution for several instances of the total costs. The detail along the left side of the above graph shows the total costs. Whereas the bottom of the graph above, shows the different classes from the instances. The chart showed the comparison between total costs obtained using the CWS algorithm and total costs obtained using IG algorithm. In summary, it is apparent that there was a slight improvement in the solutions to the instances, such as P-n76-k5. Also, a clear improvement can be seen in the results using the IG algorithm; this approach out-performs the other approach in the improvement of solutions.

§1.6 Conclusion

For the purpose of this paper, we have compared CWS with IG algorithm and then analysed the trade-offs of these solutions on instances. These two different approaches were proposed to evaluate the total cost of both approaches and to obtain efficient routing solutions for problems under deterministic cases. The main objective of this paper was to test these approaches to the well-known benchmark problem when customer demand is deterministic. Computational results were undertaken in order to analyse the performance of the CWS and IG. The computational results showed that IG performs well in comparison with the CWS of the total costs. In addition, it was observed that for tested instances, the second approach satisfies all customer demand, leading to improved solution costs. The improvement in the average deviation between these approaches was 0.53%. However, these approaches are a significantly more challenging approach to solving the problem because of the large problem size and the numbers of iterations required to obtain improved minimum costs.

§1.7 References

1. Chalghoumi, S., Ladhari, T. (2015). Iterated greedy local search and simulated annealing algorithms for the two-machine flowshop scheduling problem . International Conference On automation, Control, Engineering and Computer Science.
2. Fanjul-Peyro, L., Ruiz, R., (2010). Size-reduction heuristics for the unrelated parallel machines scheduling problem.
3. Hajer, A., & Talel, L. (2013). Iterated local search and iterated greedy local search for two machines permutation flowshop scheduling problem with time lag. IEEE.
4. Huerta-muñoz, D. L., Rios-Mercado, R, Z., Ruiz, R. (2012). An iterated greedy heuristic for a market segmentation problem with multiple attributes. Report Number: PISIS-2012-02, Affiliation: Graduate Program in Systems Engineering, UANL.
5. Mohan, R., Gopalan, N, P. (2014). Task assignment for heterogeneous computing problems using improved iterated greedy algorithm. I.J. Computer Network and Information Security. 50-55.
6. Naderi, B., Rahmani, S., & Rahmani, S. (2014). A multiobjective iterated greedy algorithm for truck scheduling in Cross-Dock problems. Hindawi Publishing Corporation, Journal of Industrial Engineering. 12 pages.
7. Naderi, B., Zandieh, M., Mohammad, S., & Fatemi, T. (2009). An iterated greedy algorithm for flexible flow lines with sequence dependent setup times to minimize total weighted Completion Time. Journal of Industrial Engineering, 3, 33–37.
8. Naderi, B., Rahmani, S., & Rahmani, S. (2014). A multiobjective iterated greedy algorithm for truck scheduling in Cross-Dock problems. Hindawi Publishing Corporation, Journal of Industrial Engineering. 12 pages.
9. Naderi, B., Zandieh, M., Mohammad, S., & Fatemi, T. (2009). An iterated greedy algorithm for flexible flow lines with sequence dependent setup times to minimize total weighted Completion Time. Journal of Industrial Engineering, 3, 33–37.

10. Pranzo, M., & Pacciarelli, D. (2015). An iterated greedy metaheuristic for the blocking job shop scheduling problem. Journal of Heuristics, 1-25.
11. Peyró, L. F., and Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. European Journal of Operational Research, 207(1), 55–69.
12. Ruiz, R., Stutzle, T. (2005). An iterated greedy algorithm for the flowshop problem with sequence dependent setup times. The 6th Metaheuristics International Conference, 817–823.
13. Ruiz, R., Stutzle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research, 177, 2033–2049.