Mehmet Akif Ersoy Üniversitesi
Sosyal Bilimler Enstitüsü Dergisi
ISSN: 1309-1387
Cilt: 8 Sayı: 17 2016 Aralık (s. 584-595)

COMPARISON OF JULIA AND MATLAB FOR SOLVING VEHICLE ROUTING PROBLEMS[1]

ARAÇ ROTALAMA PROBLEMLERİNİN ÇÖZÜMÜNDE JULİA VE MATLAB KULLANIMININ KARŞILAŞTIRILMASI

Kenan KARAGÜL[2,3], Michael G. KAY[4], Sezai TOKAT[5]

Abstract

In this study, two software projects are considered: Julog: Logistics Engineering Julia Toolbox and Matlog: Logistics Engineering Matlab Toolbox both of which can be used for solving vehicle routing problems. Then, a comparison is made with respect to their solution times. Matlab is a widely used software in engineering applications because of its user friendly interface and its set of toolboxes available for many areas of engineering applications.

Julia was developed as a dynamic scientific computing language at Massachusetts Institute of Technology (MIT) in 2013. Julia provides a REPL interface like well-known languages such as Matlab, Python, and R. The Matlog Matlab toolbox was developed as a open source toolbox at NCSU in 1996 for solving logistics engineering problems. Julog was started in 2014 as a software project that focuses on heuristic solution approaches for vehicle routing problems. The aim of this research is to compare solution times of well-known capacitated vehicle routing problem instances from the literature. A comparison is provided for Julog and Matlog functions that compute a distance matrix, determine pairwise savings, and execute the Parallel Savings Algorithm and the Two-Opt

[1]This research was presented and abstract was published in The 35th National Operations Research and Industrial Engineering Congress (ORIE 2015), September 9th through 11th, 2015 at Middle East Technical University, Ankara, Turkey.

[2]**Corresponding Author:** K. Karagul, kkaragul@pau.edu.tr
[3] Assistant Professor Dr., Pamukkale University, Logistics Department, Denizli, Turkey, kkaragul@pau.edu.tr
[4] Associate Professor Dr. , North Carolina State University, Department of Industrial and Systems Engineering, NC, USA, kay@ncsu.edu
[5] Associate Professor Dr. , Pamukkale University, Department of Computer Engineering, Denizli, Turkey, stokat@pau.edu.tr

Başvuru Tarihi: 30.10.2016
Yayına Kabul Tarihi: 02.12.2016
DOI: 10.20875/sb.02680

improvement algorithm for each instance. The results show that the Julog functions are superior to the Matlog functions for all of these well-known test instances with respect to solution time.

Keywords: Julog: Logistics Engineering Julia Toolbox, Matlog: Logistics Engineering Matlab Toolbox, Vehicle Routing Problem, Combinatorial Optimization, Julia, Matlab

Öz

Bu çalışmada her ikisi de Araç Rotalama Problemlerini çözmek için kullanılabilecek iki yazılım projesi ele alınmıştır: JuLog: Lojistik Mühendisliği Julia Araç Kutusu ve MatLog: Lojistik Mühendisliği Matlab Araç Kutusu. Daha sonra çözüm süreleri dikkate alınarak karşılaştırmalar yapılmıştır. Matlab, kullanıcı dostu arayüzü ve farklı alanlardaki mühendislik uygulamalarına ait araç kutuları ile mühendislik uygulamalarında yaygın kullanılan bir yazılımdır. Julia dinamik bilimsel hesaplama dili olarak 2013 yılında Massachusetts Teknoloji Enstitüsü'nde (MIT) geliştirilmiştir. Julia tarafından Matlab, Python ve R gibi çok yaygın dillerde de var olan REPL arayüzü bulunur. MatLog Matlab araç kutusu açık kaynak kodlu bir araç kutusu olarak 1996'da Kuzey Karolina Devlet Üniversitesi'nde (NCSU) geliştirilmiş ve lojistik mühendisliği problemlerine odaklanmıştır. JuLog 2014 yılında bir yazılım projesi olarak başlamış, rotalama problemleri ve sezgisel çözüm yaklaşımlarına yoğunlaşmıştır. Bu araştırmanın amacı yazında oldukça iyi bilinen Kapasiteli Araç Rotalama Problemi test örneklerinin çözüm sürelerinin karşılaştırılmasıdır. JuLog ve MatLog fonksiyonları ile her bir test problemi için uzaklık matrisi hesaplaması, tasarruf çiftlerinin elde edilmesi, Paralel Tasarruf Algoritması ve İki-Opt Algoritması ardışık olarak uygulanmış ve çözümler elde edilmiştir. Çözüm süreleri açısından yapılan karşılaştırmada, tüm bu test problemleri için JuLog fonksiyonları MatLog fonksiyonlarından daha başarılı sonuçlar ortaya koymuştur.

Anahtar Kelimeler: Julog: lojistik mühendisliği Julia araç kutusu, Matlog: lojistik mühendisliği Matlab araç kutusu, araç rotalama problemi, kombinatoryel optimizasyon, Julia, Matlab

1. INTRODUCTION

Vehicle routing problems represent some of the most difficult logistics-related combinatorial optimization problems. Although Matlab, a high-level language, is widely used for many logistics-related problems, the computationally intensive aspect of most vehicle routing heuristics makes it not the best choice. Instead, lower-level, more computationally efficient languages like C/C++ and Java are typically used for implementation, sometimes after first developing the interest in a high level language like Matlab or Python. Julia has the potential to provide both ease of development of a high-level language and the computational efficiency of lower-level languages, and is potentially an ideal environment for developing and implementing vehicle routing heuristics.

For researchers, especially those in the field of combinatorial optimization, programming languages are important tools for building algorithms. The efficiency and reliability of a software environment is directly related to the selected programming language. However, most researchers do not think over choosing the most recent and problem-specific programming language, and are not interested in efficiency and how to write more reliable and maintainable algorithms (Wilson et al., 2015).

Matlab is a widely used software in engineering applications. In its official web site Matlab is called "The Language of Technical Computing". It has a high-level programming language and a graphical user interface (Web-1). It is easy to learn and has a user-friendly environment and powerful graphical features that are the core reasons for its widespread usage. Julia, in its official web site, is called "A fresh approach to technical computing". It has a high-level dynamic programming language for technical computing, with syntax that is familiar to users of other technical computing environments. It provides a sophisticated compiler, distributed parallel execution, numerical accuracy, and an extensive mathematical function library (Bezanson et al., 2015). Julia was first developed at MIT in 2013 as an open-source scientific computing language and is still being actively developed by a worldwide community. Julia provides a REPL interface as in other well-known languages. It can be learned as easily as Matlab, R and Phyton, and has a performance similar to C/C++, which makes it attractive for researchers as a scientific computing tool. However, Julia does not have powerful graphical features and a user friendly graphical user interface. However, in spite of these disadvantages, the number of researchers using Julia is increasing. It is our personnel foresight that Matlab and Julia will be two important rivals of the scientific computing field in the near future. In the next section, the status of the Julia programming language in the scientific computing literature will be considered.

2. LITERATURE REVIEW

Julia programming language has been constantlydeveloped to improve its comparative advantage over other programming languages. These improvements are periodically shared at Julia official web site . In Table 1, standard test function comparisons from two different dates are given. The benchmark times in Table 1 are given relative to C programming languages, taking C performance as 1. It can be seen that the time complexity of Julia improved dramaticaly in a 9-month period. In Table 2, Matlab and Julia scientific computing languages are compared. Both Matlab and Julia have improvements with respect to their previous versions. Also, Julia has a comparative advantage with respect to other languages.

In the area of operations research, Lubin and Dunning (2015) analyze the performance of Julia, showed how Julia can be effective in linear and nonlinear optimization problems by using test problems and specialized coding procedures. In addition, in another study conducted by Dunning et al. (2015), with an operations research package known as Julia for Mathematical Programming (JuMP), the comparative advantages that can be obtained by the usage of the Julia language for the solution of linear and nonlinear programming problems as compared to commercial and other open source solvers were presented. In another study on convex optimization, it is seen that time complexity is improved by using Julia (Udell et al, 2015).

Table 1: Comparisons for Julia and other programming languages

(a) Results obtained in 10 March 2015

Test Function	Fortran gcc 4.8.2	Julia 0.3.2	Python 2.7.6	R 3.1.1	Matlab R2014a	Octave 3.8.1	Mathematica 10.0	JavaScript V8 3.14.5.9.	Go go1.2.1	LuaJIT gsl-shell 2.3.1	Java 1.7.0_65
fib	0.70	**2.39**	79.95	553.29	**4638.29**	9764.56	163.43	3.73	2.14	2.38	0.90

parse_int	4.88	**1.93**	12.24	53.23	**1580.52**	9106.83	17.66	2.33	3.77	6.79	5.55
quicksort	1.31	**1.24**	33.23	255.73	**54.43**	1766.13	48.21	2.91	1.11	2.36	1.69
mandel	0.74	**0.72**	12.18	54.06	**51.23**	391.25	6.24	1.55	0.99	0.71	0.57
pi_sum	0.99	**1.06**	16.93	16.55	**1.27**	279.53	1.51	2.19	1.33	1.18	1.00
rand_mat_stat	1.15	**2.14**	19.04	16.65	**10.48**	35.92	6.71	3.32	8.92	4.34	4.01
Rand_mat_mul	4.73	**1.11**	1.24	1.91	**1.18**	1.25	1.21	17.19	9.83	1.44	2.35

(b) Results obtained in 8 December 2015

Test Function	**Fortran gcc 5.1.1**	**Julia 0.4.0**	**Python 3.4.3**	**R 3.2.2**	**Matlab R2015b**	**Octave 4.0.0**	**Mathematica 10.2.0**	**JavaScript V8 3.28.71.19.**	**Go go1.5**	**LuaJIT gsl-shell 2.3.1**	**Java 1.8.0_45**
fib	0.70	**2.11**	77.76	533.52	**26.89**	9324.35	118.53	3.36	1.86	1.71	1.21
parse_int	5.05	**1.45**	17.02	45.73	**802.52**	9581.44	15.02	6.06	1.20	5.77	3.35
quicksort	1.31	**1.15**	32.89	264.54	**4.92**	1866.01	43.23	2.70	1.29	2.03	2.60
mandel	0.81	**0.79**	15.32	53.16	**7.58**	451.81	5.13	0.66	1.11	0.67	1.35
pi_sum	1.00	**1.00**	21.99	9.56	**1.00**	299.31	1.69	1.01	1.00	1.00	1.00
rand_mat_stat	1.45	**1.66**	17.93	14.56	**14.52**	30.93	5.95	2.30	2.96	3.27	3.92
Rand_mat_mul	3.48	**1.02**	1.14	1.57	**1.12**	1.12	1.30	15.07	1.42	1.16	2.36

(Source: Web-2)

Table 2: Comparisons for Julia and Matlab in two different versions

Test Function	**Julia 0.3.2**	**Matlab R2014a**	**Matlab/Julia Ratio**	**Julia 0.4.0**	**Matlab 2015b**	**Matlab/Julia Ratio**
Fib	2.39	4638.29	**1940.71**	2.11	26.89	**12.74**
parse_int	1.93	1580.52	**818.92**	1.45	802.52	**553.46**
Quicksort	1.24	54.43	**43.90**	1.15	4.92	**4.28**
Mandel	0.72	51.23	**71.15**	0.79	7.58	**9.59**
pi_sum	1.06	1.27	**1.20**	1.00	1.00	**1.00**
rand_mat_stat	2.14	10.48	**4.90**	1.66	14.52	**8.75**
Rand_mat_mul	1.11	1.18	**1.06**	1.02	1.12	**1.10**

In macroeconomics, Aruoba and Fernandez-Villaverde(2015) compared an algorithm of the neo-classical growth model written in C++, Fortran, Java, Julia, Python, Matlab, Mathematica and R languages in two different operating systems, Mac and Windows. Then strong and weak features of these languages are argued. In both operating systems, C++ and Fortran have the best time-complexities in both operating systems. After these two languages, Julia showed best time-complexity performance in both operating systems (Aruoba and Fernandez-Villaverde, 2015).

From the given tables and literature, it is seen that there is a trade-off between time-complexity performance and human convenience. When high-level languages are used, user friendly and easy to use environments are obtained but time-complexities are worse. Julia seems to give a new tool for scientific computing that lessens this trade-off (Bezanson et al., 2015). Also, as can be seen from the literature, there are studies on operations research and optimization areas with different programming languages and environments. However, for the vehicle routing problem, there are not any studies comparing Matlab and Julia with respect to their time-complexity performances, thus, in this study, Matlab and Julia will be examined based on their ability to solve this type of problems.

3. RESEARCH

Vehicle routing problems are challenging and quite a lot of researchers are working on this area. Therefore, a lot of heuristic solution algorithms and, to a certain extent, exact mathematical solutions, have been developed (Cordeau et al., 2002; Gendreau et al., 2007; Groer et al., 2010; Irnich et al., 2014). In our study, the test problems for the capacitated vehicle routing problem (CVRP) from the literature are used for analysis and comparison. A detailed explanation of the structure of the test problems and structural properties of the problems are given in Uchoa et al. (2014).

For the comparisons, well-known CVRP test problems such as A (27 test problems), B (23 test problems), P (24 test problems), E (5 test problems), F (3 test problems), M (5 test problems) and X (19 test problems) of (Web-5) in which 100 test problems between 100 to 1000 nodes are given. X test problems can be accessed from web page of (Web-5) and other test problems can be obtained from VRP web pages of Dorronsoro (Web-6). The A, B and P problems were introduced by Augerat et al. (1995), E test problems were introduced by Christofides and Eilon(1969), F test problems were introduced by Fisher (1994), M test problems were introduced by Christofides et al. (1979), and X test problems were introduced by Uchoa et al (2014).

For the comparison of the algorithms, the classical construction algorithm known as the Savings Algorithm is used for the solution of the CVRP. A useful discussion of the Savings Algorithm can be found in Larson and Odoni (2015). The four phases that are necessary for the solution of the CVRP problem are used for the comparisons. These phases are: calculation of the distance matrix, calculation of the savings pairs, savings-based rep construction, and a 2-opt post-improvement process. The 2-opt algorithm is discussed in Nilsson (2015), Kuang (2015), and Johnson and McGeoch (2015).

Matlog is a useful logistics engineering Matlab toolbox for implementing the CVRP algorithms. It was first developed in 1996 at NCSU, and is focused on logistics engineering problems. It is an open source toolbox for commercial Matlab (Web-3). For the CVRP algorithm implementations in Julia, Julog, a logistics engineering Julia toolbox, is used. Julog started as a new project in 2014 at NCSU and is focused on the routing problems and heuristics solution approaches (Web-4). The comparisons are implemented using Matlab version 2013a and Julia version 0.2.8. Windows 8 operating system, Intel (R) Core(TM) i7-4800MQ 2.70 GHz CPU and 16 GB RAM with 64-bit computer. Only one core of the CPU is used for the simulations.

In Table 3, average solution times of CVRP test problems are given. In this table, distance means distance matrix calculation time, pairwise means savings pairs calculation time, savings means savings routing function calculation time, and TwoOpt (2-opt route improvement algorithm) run time all given in seconds. In addition, the ratio of the aforementioned values for Matlog and Julog are given under MatlogTime/JulogTime.

For each group of A, B, P, E-F-M and X test instances summarized in Table3, detailed results are shown in Appendices 1–5, respectively. All of the measurements are in seconds. In distance matrix calculations, Julog functions are a maximum of 13 times and a minimum of 0.43 times faster than the corresponding Matlog functions. In pairwise calculations of the Savings Algorithm, functions written in Julog are 2.5–3 times faster than the Matlog

functions. Also, in the route constructions of the Savings algorithm, Julog is 2 times faster on average. The 2-opt routing improvement function is at least 5 times faster than Matlog.

It is an interesting fact that as matrix dimensions of the distance matrix calculations of Julog increase, the advantage of it calculation speed becomes less. However, it is not shown here but the statistical package of Julia's built-in distance function is at least 2 times faster than the Matlog distance function for all distance matrix dimensions. Therefore, for distance matrix calculations in Julog, new code is developed that uses for-next loop calculations instead of matrix calculations for obtaining distance matrix calculations.

Table 3: Comparisons Test Instances Group Averages Solution Times for Matlog/Matlab and Julog/Julia

Instances	Matlab-MatLog Calculation Times (Seconds)				Julia-JuLog Calculation Times (Seconds)				MatlogTime/JuLogTime			
Group	Distance	Pairwise	Savings	TwoOpt	Distance	Pairwise	Savings	TwoOpt	Distance	Pairwise	Savings	TwoOpt
A	0.001351	3.401156	5.487842	0.713018	0.000136	1.189609	3.023731	0.132019	**9.96**	**2.86**	**1.81**	**5.40**
B	0.001544	3.569580	4.018624	0.676913	0.000131	1.212898	2.148238	0.118110	**11.76**	**2.94**	**1.87**	**5.73**
P	0.001719	3.665243	9.276295	1.845574	0.000135	1.244313	5.708650	0.328465	**12.75**	**2.95**	**1.62**	**5.62**
E	0.001094	6.801875	14.238120	2.053106	0.000215	2.290731	8.109390	0.358056	**5.08**	**2.97**	**1.76**	**5.73**
F	0.001331	11.866159	97.330308	18.478971	0.000407	3.975892	67.726119	2.105337	**3.27**	**2.98**	**1.44**	**8.78**
M	0.001352	37.436539	66.317247	6.953927	0.001161	12.591500	38.075558	1.095581	**1.16**	**2.97**	**1.74**	**6.35**
X	0.016872	835.345180	1698.253676	43.481111	0.039661	331.167361	955.190306	7.681458	**0.43**	**2.52**	**1.78**	**5.66**

The results obtained by using the new distance function are given in Table 4, where the X problems that are analyzed are only composed of large instances. The loop-based solutions of Julog are considered as JulogNewDist in Table 4. For the loop based case, the new ratio is given under NewRatio in the table. It can be seen that by using loop based distance calculations, Julia is again faster. On average, the time complexity of Julog is 5 times better than Matlog, has can be observed in Table 4.

In this study, the test problems of classical algorithms given by the main Julia team and related research in the literature are investigated. Also, for problems with high algorithm complexity, Julia and Matlab scientific computations tools are analyzed by using the CVRP as the benchmark tool. It is seen that Julia is a powerful open-source scientific computing language.

Table 4: New Solution Approach for Distance Matrix Calculation Times (seconds)

	Instance	BKS	MatLog	JuLogNewDists	NewRatio Matlog/NewJ	OldRatio Matlog/NewJ
1	X-n101-k25	27591	0.013508	0.000127726	105.76	89.82
2	X-n157-k13	16876	0.013195	0.000326156	40.46	21.41
3	X-n200-k36	58578	0.013708	0.000541693	25.31	0.80
4	X-n251-k28	38684	0.013797	0.000828315	16.66	0.91
5	X-n303-k21	21744	0.029644	0.001805264	16.42	0.43
6	X-n351-k40	25946	0.014984	0.001627742	9.21	0.48
7	X-n401-k29	66243	0.018430	0.002359883	7.81	0.56
8	X-n459-k26	24181	0.018222	0.002828970	6.44	0.42
9	X-n502-k39	69253	0.021054	0.002567817	8.20	0.21
10	X-n561-k42	42756	0.022293	0.004134356	5.39	0.21
11	X-n613-k62	59778	0.025434	0.004933782	5.16	0.26
12	X-n655-k131	106780	0.027501	0.008610455	3.19	0.28
13	X-n701-k44	82292	0.029131	0.006489298	4.49	0.33
14	X-n766-k71	114683	0.032612	0.007788602	4.19	0.27
15	X-n801-k40	73587	0.033258	0.008548874	3.89	0.36
16	X-n856-k95	89060	0.036146	0.010930424	3.31	0.32
17	X-n916-k207	329836	0.039525	0.011583878	3.41	0.38
18	X-n957-k87	85672	0.040084	0.013000011	3.08	0.40
19	X-n1001-k43	72742	0.048646	0.013000011	3.74	0.37
Average					**14.53**	**6.22**
Maximum					**105.76**	**89.82**
Minimum					**3.08**	**0.21**
Median					**5.39**	**0.38**

4. CONCLUSION AND DISCUSSION

In scientific computation languages, Matlab is a well-known, widespread tool, while Julia is a new open-source scientific computing language with and accelerating number of users worldwide since it was announced in 2013. Because of its being open source, Julia has a huge volunteer support worldwide. Thus, various packages have been developed for Julia and the language itself is continuously evolving. In this study, the four solution phases of the vehicle routing problem are considered: the distance matrix calculation, obtaining the Savings pairs as a preparation for the Savings route construction algorithm, the savings route construction algorithm, and finally the 2-opt post optimization algorithm. Those four calculation phases are analyzed in terms of time complexities and Julia and Matlab scientific computing languages are compared based on the Julog and Matlog toolboxes that have been developed for each language, respectively. As a result of the comparison, it is seen that Julog has a better calculation time performance than Matlog. Compared with Matlab, it is seen from the performance analyses that Julia is on average five times better in distance matrix calculations, three times better in savings pair calculations, two times better in savings routing algorithms, and five times better in 2-opt routing post-optimization algorithm. Therefore, the Julia scientific computing language is suitable for combinatorial optimization problems, and can be expected in the near future to reach a performance in the same class with C/C++ while providing the same type of user-friendly interface found only today and languages such as Matlab, Python and R.

ACKNOWLEDGEMENTS: This project was partially supported by the Turkish Council of Higher Education and The Scientific and Technological Research Council of Turkey. The first author would like to thank the NCSU

Industrial and Systems Engineering Department and Operations Research Graduate Program for their hospitality during the initial writing of this paper.

5. REFERENCES

Aruoba, S. Borağan - Fernandez-Villaverde, Jesus (2015), "A Comparison of Programming Languages in Macroeconomics", *Journal of Economic Dynamics and Control*, Volume.58, (265-273).

Bezanson, Jeff - Edelman, Alan - Karpinski, Stefan - Shah, Viral B. (2015), "Julia. Julia Language", http://julialang.org (08.12.2015)

Bezanson, Jeff - Edelman, Alan - Karpinski, Stefan - Shah, Viral B. (2014), "Julia: A fresh approach to numerical computing", https://arxiv.org/abs/1411.1607v4 (28.12.2016)

Cordeau, Jean-François - Gendreau, Michel - Laporte, Gilbert - Potvin, Jean-Yves-Semet, Frédéric (2002), "A guide to vehicle routing heuristics", *The Journal of the Operational Research Society*, Volume.53, Issue.5, (512-522).

CVRPLIB (2015), "Capacitated Vehicle Routing Problem Library", http://vrp.galgos.inf.puc-rio.br/index.php/en/ (10.12.2015)

Dorronsoro, Bernabé. (2015), "VRP Instances. The VRP Web", http://www.bernabe.dorronsoro.es/vrp/ (10.12.2015)

Dunning, Iain - Gupta, Vishal - King, Angela - Kung, Jerry - Lubin, Miles - Silberholz, John (2014), "A Course on Advanced Software Tools for Operations Research and Analytics", *INFORMS Transactions on Education*, Volume.15, Issue.2, (169-179).

Dunning, Iain-Huchette, Joey-Lubin, Miles (2015), "JuMP: A Modelling Language for Mathematical Optimization", https://arxiv.org/abs/1508.01982 (10.12.2015)

Gendreau, Michel-Potvin, Jean-Yves-Braysy, Olli-Hasle, Geir-Lokketangen, Arne (2008), "Metaheuristic for the Vehicle Routing Problem and its Extensions: A Categorized Bibliography", (Ed) B. Golden, S. Raghavan and E. Wasil, *The Vehicle Routing Problem: Latest Advances and New Challenges*, Volume 43 of the series Operations Research/Computer Science Interfaces, (143-169).

Groer, Chris-Golden, Bruce-Wasil, Edward (2010), "A Library of Local Search Heuristics for the Vehicle Routing Problem", *Mathematical Programming Computation*, Volume.2, Issue.2, (79-101).

Irnich, Stefan, Toth, Paolo-Vigo, Daniele (2014), "The family of vehicle routing problems", (Ed) Paolo Toth and Daniele Vigo, The Vehicle Routing Problems, Methods, and Applications, 2nd edition, Philadelphia: SIAM, USA, (1-26).

Johnson, David S. - McGeoch, Lyle A. (1997), "The Travelling Salesman Problem: A Case Study in Local Optimization", (Ed) Aarts, Emile - Lenstra, Jan Karel *Local Search in Combinatorial Optimization*, John Wiley & Sons, Chichester, UK, (215–310).

Karagul, Kenan-Kay, Michael G. (2015), "Julog: Logistics Engineering Julia Toolbox", http://www.ise.ncsu.edu/kay/Julog/index.html (14.12.2015)

Kay, Michael G. (2015), "Matlog: Logistics Engineering Matlab Toolbox", http://www.ise.ncsu.edu/kay/Matlog/index.htm (14.12.2015)

Kuang, Eric (2012), *A 2-opt-based Heuristic for the Hierarchical Traveling Salesman Problem*, Undergraduate Thesis, Honors Graduates, Advisor: B. Golden, Department of Computer Science, University of Maryland.

Larson, Richard C.- Odoni, Amedo R. (1981), Chapter 6: Network Applications, *In* Urban Operations Research, http://web.mit.edu/urban_or_book/www/book/chapter6/contents6.html (10.12.2015)

Lubin, Miles-Dunning, Iain (2015), "Computing in Operations Research using Julia", *INFORMS Journal on Computing*, Volume.27, Issue.2, (238-248).

Mathworks (2015), "The Language of Technical Computing", http://www.mathworks.com/products/matlab/ (08.12.2015)

Nilsson, Christian (2003), *Heuristics for the Traveling Salesman Problem*, Technical Report, Linköping University, Sweden.

Uchoa, Eduardo-Pecin, Diego-Pessoa, Artur-Poggi, Marcus-Subramanian, Anand-Vidal, Thibaut (2014), "New Benchmark Instances for the Capacitated Vehicle Routing Problem", ROUTE'14, June 1-4, Snekkerstern, Denmark.

Udell, Madeleine-Mohan, Karanveer-Zeng, David-Hong, Jenny-Diamond, Steven-Boyd, Stephen (2014), "Convex Optimization in Julia, First Workshop for High Performance Technical Computing in Dynamic Languages", November 16-21, New Orleans, Louisiana, USA.

Wilson, Greg-Aruliah, Dhavide A.-Brown, C. Titus-Chue Hong, Neil. P.-Davis, Matt-Guy, Richard T.- et al (2014), "Best Practices for Scientific Computing", *PLoS Biology*, Volume.12, Issue.1, (1-7).

Appendix

Appendix 1: Test Instances A:Solution times for Matlog/Matlab and Julog/Julia

			Matlab-MatLog Calculation Times (Seconds)				Julia-JuLog Calculation Times (Seconds)				MatlogTime/JuLogTime			
	Instance	BKS	Distance	Pairwise	Savings	TwoOpt	Distance	Pairwise	Savings	TwoOpt	Distance	Pairwise	Savings	TwoOpt
1	A-n32-k5	784	0.001286	1.318264	2.149545	0.391086	6.652E-05	0.445465	1.160648	0.067712	19.33	2.96	1.85	5.78
2	A-n33-k5	661	0.001276	1.335570	1.689015	0.328733	7.261E-05	0.488775	0.885719	0.052813	17.57	2.73	1.91	6.22
3	A-n33-k6	742	0.000876	1.344495	2.073850	0.282368	5.93E-05	0.460681	1.10111	0.085487	14.77	2.92	1.88	3.30
4	A-n34-k5	778	0.002151	1.431419	1.907757	0.335466	7.641E-05	0.506286	1.031463	0.071873	28.15	2.83	1.85	4.67
5	A-n36-k5	799	0.000878	1.607190	2.747690	0.550597	8.857E-05	0.568321	1.496771	0.101377	9.91	2.83	1.84	5.43
6	A-n37-k5	669	0.000981	1.713375	3.014082	0.850176	0.0002387	0.635817	1.689197	0.120578	4.11	2.69	1.78	7.05
7	A-n37-k6	949	0.000840	1.715535	3.231237	0.481331	8.439E-05	0.60871	1.741842	0.077276	9.95	2.82	1.86	6.23
8	A-n38-k5	730	0.001304	1.805683	2.263773	0.376951	7.413E-05	0.637178	1.228728	0.075179	17.59	2.83	1.84	5.01
9	A-n39-k5	822	0.001709	1.945516	3.876666	0.649717	0.0001015	0.67894	2.189239	0.218083	16.84	2.87	1.77	2.98
10	A-n39-k6	831	0.001511	1.905682	3.146009	0.576612	9.503E-05	0.680949	1.714596	0.099392	15.90	2.80	1.83	5.80
11	A-n44-k6	937	0.000895	2.462617	4.366901	0.547487	0.0001266	0.867718	2.346607	0.086765	7.07	2.84	1.86	6.31
12	A-n45-k6	944	0.001349	2.586699	3.174293	0.448617	0.0001007	0.907715	1.701916	0.077656	13.39	2.85	1.87	5.78
13	A-n45-k7	1146	0.001735	2.558271	3.948853	0.475006	0.0001087	0.889472	2.091344	0.071343	15.96	2.88	1.89	6.66
14	A-n46-k7	914	0.000912	2.689433	4.123364	0.701236	0.0001175	0.960942	2.246224	0.100907	7.76	2.80	1.84	6.95
15	A-n48-k7	1073	0.001594	2.967503	5.057367	0.680872	0.0001038	1.041609	2.809756	0.129032	15.36	2.85	1.80	5.28
16	A-n53-k7	1010	0.001508	3.632820	7.727971	1.002483	0.0001399	1.265399	4.305179	0.187955	10.78	2.87	1.80	5.33
17	A-n54-k7	1167	0.000951	3.865395	8.350985	1.252973	0.0001372	1.313219	4.611502	0.270229	6.93	2.94	1.81	4.64
18	A-n55-k9	1073	0.001003	3.932658	3.846772	0.625447	0.0001372	1.384533	2.015325	0.09922	7.31	2.84	1.91	6.30
19	A-n60-k9	1354	0.001479	4.698968	6.300114	0.863571	0.0001733	1.607118	3.384089	0.147066	8.53	2.92	1.86	5.87
20	A-n61-k9	1034	0.001771	5.023285	5.457696	0.759876	0.0001654	1.700479	2.82519	0.144983	10.71	2.95	1.93	5.24
21	A-n62-k8	1288	0.001678	5.183572	10.386439	1.262806	0.0001714	1.756014	5.721486	0.207871	9.79	2.95	1.82	6.07
22	A-n63-k9	1616	0.001017	5.192808	8.459120	0.873141	0.0001768	1.845392	4.674984	0.12774	5.75	2.81	1.81	6.84
23	A-n63-k10	1314	0.000977	5.189668	7.103424	0.859383	0.0001787	1.828323	3.935454	0.151227	5.47	2.84	1.80	5.68
24	A-n64-k9	1401	0.001193	5.351454	11.682432	1.095978	0.0001847	1.884751	6.725359	0.189674	6.46	2.84	1.74	5.78
25	A-n65-k9	1174	0.001314	5.525694	5.952502	0.760801	0.0001851	1.959936	3.207546	0.150471	7.10	2.82	1.86	5.06
26	A-n69-k9	1159	0.002491	6.288865	9.094454	0.789611	0.0002136	2.202134	5.023058	0.166894	11.66	2.86	1.81	4.73
27	A-n80-k10	1763	0.001791	8.558786	17.039424	1.42916	0.0002851	2.993577	9.776392	0.285702	6.28	2.86	1.74	5.00
Averages		**1041.93**	**0.001351**	**3.40**	**5.49**	**0.71**	**0.000136**	**1.19**	**3.02**	**0.13**	**9.96**	**2.86**	**1.81**	**5.40**

Appendix 2: Test Instances B: Solution times for Matlog of Matlab and Julog of Julia

			Matlab-MatLog Calculation Times (Seconds)				Julia-JuLog Calculation Times (Seconds)				MatlogTime/JuLogTime			
	Instance	BKS	Distance	Pairwise	Savings	TwoOpt	Distance	Pairwise	Savings	TwoOpt	Distance	Pairwise	Savings	TwoOpt
1	B-n31-k5	672	0.001467	1.199152	1.287563	0.340838	5.816E-05	0.407125	0.654273	0.07737	25.22	2.95	1.97	4.41
2	B-n34-k5	788	0.001931	1.438258	1.421893	0.398993	7.071E-05	0.485697	0.721408	0.060487	27.31	2.96	1.97	6.60
3	B-n35-k5	955	0.000934	1.522128	1.358583	0.605441	6.804E-05	0.513126	0.777786	0.10957	13.73	2.97	1.75	5.53
4	B-n38-k6	805	0.001652	1.798031	1.843832	0.417401	7.299E-05	0.61623	0.980237	0.084856	22.63	2.92	1.88	4.92
5	B-n39-k5	549	0.001644	1.907146	2.609530	0.867351	9.085E-05	0.654319	1.512482	0.124848	18.10	2.91	1.73	6.95
6	B-n41-k6	829	0.001682	2.116596	2.269504	0.273558	9.123E-05	0.734965	1.148819	0.05933	18.44	2.88	1.98	4.61
7	B-n43-k6	742	0.001458	2.333909	3.573092	0.641608	9.541E-05	0.805614	1.855935	0.077113	15.28	2.90	1.93	8.32
8	B-n44-k7	909	0.001595	2.469156	3.435483	0.556592	9.465E-05	0.831067	1.835517	0.079623	16.85	2.97	1.87	6.99
9	B-n45-k5	751	0.001681	2.605426	3.550059	1.053197	9.579E-05	0.875601	1.886142	0.145041	17.55	2.98	1.88	7.26
10	B-n45-k6	678	0.001226	2.599443	3.445221	0.730547	0.0001042	0.877689	1.833129	0.111608	11.77	2.96	1.88	6.55
11	B-n50-k7	741	0.001732	3.213264	3.347693	0.623676	0.0001376	1.096394	1.835189	0.119897	12.59	2.93	1.82	5.20
12	B-n50-k8	1312	0.000947	3.198565	4.974597	0.397923	0.0001232	1.078791	2.576541	0.071106	7.69	2.96	1.93	5.60
13	B-n51-k7	1032	0.001526	3.324261	2.314448	0.649592	0.0001243	1.122791	1.296471	0.145843	12.28	2.96	1.79	4.45
14	B-n52-k7	747	0.001676	3.474931	2.999625	0.609897	0.0001228	1.197344	1.621104	0.116543	13.65	2.90	1.85	5.23
15	B-n56-k7	707	0.001782	4.062339	5.040865	1.001917	0.0001589	1.361089	2.754977	0.178698	11.21	2.98	1.83	5.61
16	B-n57-k7	1153	0.001883	4.221432	3.899267	0.687478	0.0001574	1.426716	2.063685	0.094978	11.96	2.96	1.89	7.24
17	B-n57-k9	1598	0.001239	4.214710	5.783630	0.569881	0.0001543	1.427879	3.059693	0.109458	8.03	2.95	1.89	5.21
18	B-n63-k10	1496	0.001030	5.287444	5.224365	0.641714	0.0001703	1.756128	2.754873	0.118757	6.05	3.01	1.90	5.40
19	B-n64-k9	861	0.001310	5.367139	4.544414	0.722348	0.0002186	1.823041	2.449777	0.164224	5.99	2.94	1.86	4.40
20	B-n66-k9	1316	0.001070	5.681936	7.795680	0.749931	0.0001475	1.958782	4.229716	0.154051	7.25	2.90	1.84	4.87
21	B-n67-k10	1032	0.001834	5.895207	5.616632	0.769723	0.0001893	2.007254	2.96058	0.158706	9.69	2.94	1.90	4.85
22	B-n68-k9	1272	0.001770	6.068330	6.588124	0.90525	0.0002098	2.08373	3.506256	0.180944	8.44	2.91	1.88	5.00
23	B-n78-k10	1221	0.002434	8.101541	9.504254	1.354138	0.0002619	2.755273	5.094875	0.173483	9.29	2.94	1.87	7.81
Averages		**963.74**	**0.001544**	**3.57**	**4.02**	**0.68**	**0.000131**	**1.21**	**2.15**	**0.12**	**11.76**	**2.94**	**1.87**	**5.73**

Appendix 3: Test Instances P: Solution times for Matlog of Matlab and Julog of Julia

			Matlab-MatLog Calculation Times (Seconds)				Julia-JuLog Calculation Times (Seconds)				MatlogTime/JuLogTime			
	Instance	BKS	Distance	Pairwise	Savings	TwoOpt	Distance	Pairwise	Savings	TwoOpt	Distance	Pairwise	Savings	TwoOpt
1	P-n16-k8	450	0.000991	0.281067	0.156685	0.021167	2.357E-05	0.116281	0.067818	0.003416	42.05	2.42	2.31	6.20
2	P-n19-k2	212	0.001706	0.420693	1.013673	0.221096	3.269E-05	0.147972	0.557366	0.056792	52.18	2.84	1.82	3.89
3	P-n20-k2	216	0.001953	0.473181	1.308366	0.582123	4.143E-05	0.17677	0.724598	0.058828	47.14	2.68	1.81	9.90
4	P-n21-k2	211	0.002201	0.51476	1.458819	0.597423	4.296E-05	0.191069	0.816065	0.055855	51.24	2.69	1.79	10.70
5	P-n22-k2	216	0.001542	0.568005	1.642631	0.692530	4.409E-05	0.206156	0.919899	0.059435	34.97	2.76	1.79	11.65
6	P-n22-k8	603	0.002013	0.570797	0.438065	0.047672	3.991E-05	0.199444	0.201084	0.030055	50.43	2.86	2.18	1.59
7	P-n23-k8	529	0.001030	0.624017	0.567097	0.050765	5.018E-05	0.236011	0.251422	0.030586	20.53	2.64	2.26	1.66
8	P-n40-k5	458	0.002620	2.025389	3.820672	1.007098	8.857E-05	0.675797	1.96968	0.119081	29.58	3.00	1.94	8.46
9	P-n45-k5	510	0.001529	2.568016	4.240123	1.226593	0.0001057	0.889139	2.393847	0.155776	14.47	2.89	1.77	7.87
10	P-n50-k7	554	0.001765	3.244400	3.889823	0.480718	0.0001258	1.077267	2.056623	0.140085	14.03	3.01	1.89	3.43
11	P-n50-k8	631	0.001782	3.214788	3.194508	0.383964	0.0001224	1.105201	1.628836	0.090928	14.56	2.91	1.96	4.22
12	P-n50-k10	696	0.001863	3.198331	2.823049	0.355033	0.0001209	1.085684	1.431065	0.063548	15.41	2.95	1.97	5.59
13	P-n51-k10	741	0.001412	3.362426	3.107720	0.472976	0.0001281	1.133784	1.57718	0.068855	11.02	2.97	1.97	6.87
14	P-n55-k7	568	0.001062	3.931192	5.468617	0.885919	0.0001323	1.339156	2.897612	0.168101	8.03	2.94	1.89	5.27
15	P-n55-k8	588	0.002267	3.949505	5.019547	1.200786	0.0001399	1.345724	2.647404	0.165726	16.21	2.93	1.90	7.25
16	P-n55-k10	694	0.001842	3.915118	3.460411	0.438817	0.0001893	1.330315	1.724633	0.094859	9.73	2.94	2.01	4.63
17	P-n55-k15	989	0.001889	3.918321	2.685686	0.163441	0.0001407	1.310967	1.267607	0.047895	13.43	2.99	2.12	3.41
18	P-n60-k10	744	0.001443	4.676873	4.783961	0.769251	0.0001441	1.590305	2.430018	0.1525	10.02	2.94	1.97	5.04
19	P-n60-k15	968	0.001721	4.802095	3.252546	0.265655	0.0001612	1.583045	1.592073	0.06096	10.68	3.03	2.04	4.36
20	P-n65-k10	792	0.002035	5.568511	6.271030	0.798730	0.0001828	1.901445	3.272308	0.160101	11.13	2.93	1.92	4.99
21	P-n70-k10	827	0.001695	6.574622	6.845599	1.047527	0.0002041	2.224013	3.597033	0.211412	8.30	2.96	1.90	4.95
22	P-n76-k4	593	0.001613	7.778138	38.280404	7.220931	0.0002927	2.627847	23.99772	0.686934	5.51	2.96	1.60	10.51
23	P-n76-k5	627	0.001728	7.70538	26.525160	3.587132	0.0002532	2.616055	16.09818	0.880815	6.83	2.95	1.65	4.07
24	P-n101-k4	681	0.001562	14.080201	92.376876	21.77644	0.0004296	4.754059	62.88753	4.320614	3.64	2.96	1.47	5.04
Averages		**587.42**	**0.001719**	**3.67**	**9.28**	**1.85**	**0.000135**	**1.24**	**5.71**	**0.33**	**12.75**	**2.95**	**1.62**	**5.62**

Appendix 4: Test Instances E, F and M: Solution times for Matlog of Matlab and Julog of Julia

			Matlab-MatLog Calculation Times (Seconds)				Julia-JuLog Calculation Times (Seconds)				MatlogTime/JuLogTime			
	Instance	BKS	Distance	Pairwise	Savings	TwoOpt	Distance	Pairwise	Savings	TwoOpt	Distance	Pairwise	Savings	TwoOpt
1	E-n30-k3	534	0.000990	1.140989	2.225799	1.37877	4.98E-05	0.37729	1.231072	0.105721	19.88	3.02	1.81	13.04
2	E-n51-k5	521	0.001197	3.431971	7.616093	1.21673	0.0001308	1.127738	4.18879	0.232021	9.15	3.04	1.82	5.24
3	E-n76-k8	735	0.001498	7.746285	14.703431	1.89599	0.0002429	2.601536	8.168994	0.504974	6.17	2.98	1.80	3.75
4	E-n76-l10	830	0.000860	7.736440	10.164623	1.01052	0.0002456	2.607138	5.466729	0.311693	3.50	2.97	1.86	3.24
5	E-n101-k8	817	0.000926	13.95369	36.480655	4.76353	0.0004071	4.739953	21.49136	0.635871	2.27	2.94	1.70	7.49
Averages		**687.40**	**0.001094**	**6.80**	**14.24**	**2.05**	**0.000215**	**2.29**	**8.11**	**0.36**	**5.08**	**2.97**	**1.76**	**5.73**
1	F-n45-k4	724	0.001400	2.60045	9.594766	2.00775	0.0001524	0.878151	5.286539	0.203321	9.18	2.96	1.81	9.87
2	F-n72-k4	237	0.001243	7.06309	47.221996	8.30568	0.000211	2.350508	32.43527	1.946387	5.89	3.00	1.46	4.27
3	F-n135-k7	1162	0.001349	25.93493	235.17416	45.12348	0.0008568	8.699017	165.4565	4.166303	1.57	2.98	1.42	10.83
Averages		**707.67**	**0.001331**	**11.87**	**97.33**	**18.48**	**0.000407**	**3.98**	**67.73**	**2.11**	3.27	2.98	1.44	8.78
1	M-n101-k10	820	0.001121	14.25297	19.376903	2.88405	0.0004124	4.758914	11.03233	0.353331	2.72	3.00	1.76	8.16
2	M-n121-k7	1034	0.001166	20.82446	62.162583	12.39696	0.0005603	6.889522	37.73284	1.541719	2.08	3.02	1.65	8.04
3	M-n151-k12	1053	0.001599	32.85783	62.717869	6.39306	0.0010408	11.06368	35.91747	1.030191	1.54	2.97	1.75	6.21
4	M-n200-k16	1373	0.001410	59.54834	93.558027	6.55419	0.0018037	20.17314	52.64558	1.260427	0.78	2.95	1.78	5.20
5	M-n200-k17	1373	0.001464	59.69909	93.770853	6.54139	0.0019892	20.07225	53.04957	1.292237	0.74	2.97	1.77	5.06
Averages		**1130.60**	**0.001352**	**37.44**	**66.32**	**6.95**	**0.001161**	**12.59**	**38.08**	**1.10**	**1.16**	**2.97**	**1.74**	**6.35**

Appendix 5: Test Instances X: Solution times for Matlog of Matlab and Julog of Julia

		Matlab-MatLog Calculation Times (Seconds)				Julia-JuLog Calculation Times (Seconds)				MatlogTime/JuLogTime			
	Instance	Distance	Pairwise	Savings	TwoOpt	Distance	Pairwise	Savings	TwoOpt	Distance	Pairwise	Savings	TwoOpt
1	X-n101-k25	0.048312	15.340677	13.000730	0.582806	0.000537891	4.982808	6.284019	0.107686	89.82	3.08	2.07	5.41
2	X-n157-k13	0.026169	38.357456	64.641460	5.876797	0.001222514	12.92672	36.51933	1.096866	21.41	2.97	1.77	5.36
3	X-n200-k36	0.001921	64.027343	54.091657	1.613261	0.002416139	21.96151	27.47773	0.349172	0.80	2.92	1.97	4.62
4	X-n251-k28	0.002436	103.960030	131.460295	6.108568	0.002690596	35.73979	69.6565	1.15964	0.91	2.91	1.89	5.27
5	X-n303-k21	0.002343	155.964836	345.963582	23.298037	0.005494092	54.06819	201.6471	3.97387	0.43	2.88	1.72	5.86
6	X-n351-k40	0.003463	216.092689	287.780247	9.512027	0.007248415	74.92695	152.2994	1.370918	0.48	2.88	1.89	6.94
7	X-n401-k29	0.005586	289.836743	1653.718894	50.553188	0.009947755	101.9441	980.4536	7.005026	0.56	2.84	1.69	7.22
8	X-n459-k26	0.010168	397.599906	1777.137144	51.169996	0.023993365	139.4556	1045.099	11.75277	0.42	2.85	1.70	4.35
9	X-n502-k39	0.009564	486.504472	661.940255	29.216078	0.045140277	163.4429	345.6642	5.340106	0.21	2.98	1.91	5.47
10	X-n561-k42	0.010844	628.395487	753.008272	37.052301	0.051588888	224.6824	403.64	5.770079	0.21	2.80	1.87	6.42
11	X-n613-k62	0.012834	766.545905	706.810981	26.019269	0.049657042	281.1895	352.3744	3.74273	0.26	2.73	2.01	6.95
12	X-n655-k131	0.014553	888.171424	475.047924	5.812893	0.051977766	329.5699	211.4006	1.104345	0.28	2.69	2.25	5.26
13	X-n701-k44	0.018184	1,063.600938	2767.150467	75.830070	0.055050398	390.3115	1531.168	13.1439	0.33	2.73	1.81	5.77
14	X-n766-k71	0.021166	1,347.569868	4834.284340	84.496731	0.07968125	496.414	2687.06	12.57778	0.27	2.71	1.80	6.72
15	X-n801-k40	0.021778	1,432.172650	4394.326480	122.00786	0.061290315	551.8792	2627.142	21.44489	0.36	2.60	1.67	5.69
16	X-n856-k95	0.026997	1,728.003812	931.166318	26.10205	0.085026328	669.6043	478.0467	4.774306	0.32	2.58	1.95	5.47
17	X-n916-k207	0.027699	2,019.119465	1145.067096	7.07234	0.072950958	806.1639	496.3254	1.463881	0.38	2.50	2.31	4.83
18	X-n957-k87	0.027981	2,010.614325	2017.111400	46.71580	0.070052048	907.5748	1072.663	7.796922	0.40	2.22	1.88	5.99
19	X-n1001-k43	0.028564	2,219.680402	9253.112299	217.10104	0.077592787	1025.342	5423.694	41.97283	0.37	2.16	1.71	5.17
	Average	**0.016872**	**835.35**	**1698.25**	**43.48**	**0.03966099**	**331.17**	**955.19**	**7.68**	**0.43**	**2.52**	**1.78**	**5.66**