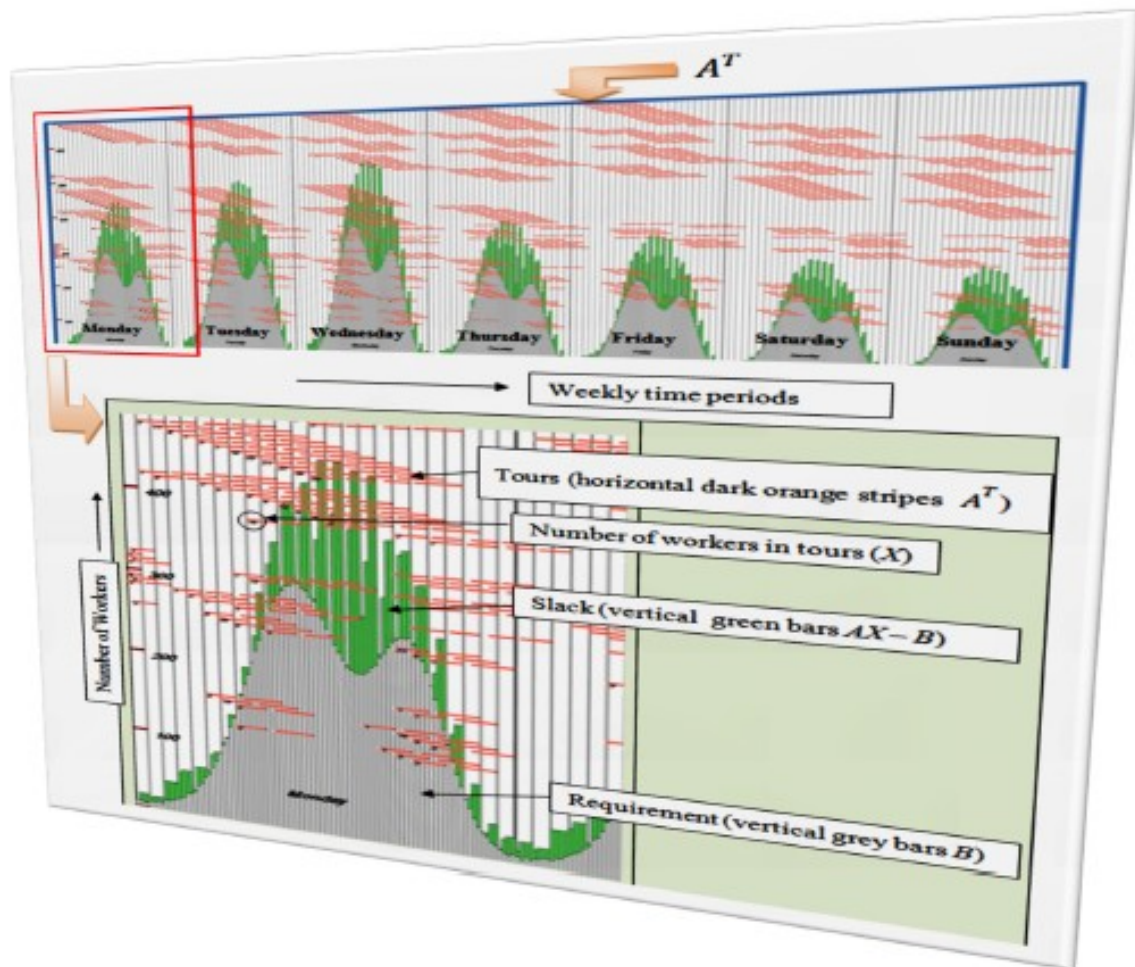


Dantzig's Workforce Scheduling Model

A spreadsheet solution for call centers using public solvers



Manjini Saminathan PhD Bahiru Kassahun PhD Arun K. Singh PhD

Dantzig's Workforce Scheduling Model

A spreadsheet solution for call centers using public solvers

A practical guide for call center analysts to implement Dantzig's seminal workforce scheduling model to generate, evaluate, and optimize service level performance of weekly agent schedules using Microsoft Excel 2007, C#.NET components and publicly available solvers, algorithms, and hardware.

Manjini Saminathan PhD Bahiru Kassahun PhD Arun Singh PhD

© 2008 Manjini Saminathan
All rights reserved.

ISBN
LCCN

Printed in the United States of America

Preface

One of the major challenges that call centers face today is to operate the center in such a way that its weekly inter- and intra-day variations in the projected call load can be handled at a prescribed service level to its customers with available agent resources at the least possible cost. This can be accomplished by scheduling their agents appropriately, so that enough agents are assigned in each time period during the week to handle the offered load without jeopardizing service quality. A number of commercial software vendors make such a scheduling tool available as part of an overall proprietary workforce management system. A great number of researchers have published on this subject in scholarly journals. These public resources do not often make software tools available to a curious analyst who may have the need to experiment with the published models and algorithms in order to understand potential benefits, if any, that can be derived for his/her center. Also, once a schedule has been generated, a simple to use software tool is not readily available to run the center in a virtual mode using a stochastic simulator. Such an environment will allow the schedule to be exercised prior to deployment using random call arrivals depicting the incoming call load. The effects on service level under expected load and/or unexpected surge conditions can be understood at any time interval during the week. A potential outcome would be to derive a quantitative estimate of the cost and resource requirements needed to provide a given level of service to the customer.

This book bridges the gap that exists today between potentially good published models in the scheduling arena and a great number of software tools to solve such models, all available in the public domain. In particular, the seminal model proposed in 1954 by Prof. Dantzig [1], who invented Linear Programming and the Simplex algorithm seven years earlier, will be exclusively used to illustrate scheduling of agents in a call center. This original model to schedule workers subjected to varying workload requirements at different time periods during a given day laid the groundwork for a number of published call center scheduling models [7, 18, 19, 20, 21, 22, 26, 40] in the literature. In our opinion, this classic model in its original form provides most capabilities needed by a user to design a research tool and conduct a call center scheduling and optimization analysis.

A number of software tools are available in the public domain to solve Dantzig's model. In particular, we will focus on five of the solvers, namely, GLPK [2], lp_solve [3], COIN-OR Cbc [41] and NEOS(FEASPUMP and SCIP) [4] that implement Dantzig's algorithm in some shape or form to obtain a solution of the model in which some or all of the variables can be defined to be integer and/or binary. In our experience, these solvers are capable of handling models that can be encountered in scheduling large call centers employing thousands of agents within a reasonable amount of time using inexpensive and powerful desktop/laptop available in the market today. As of this writing the NEOS sys-

tem even allows the user to submit the model over the internet to be solved on a publicly available hardware cluster and to retrieve the result on line or on demand. The capabilities of these solvers to obtain a schedule for a relatively simple case versus a difficult one involving binary decision variables to implement “either-or” constraints will be discussed. Practical considerations may dictate that the tours in a schedule be staffed at or above a minimum number and sometimes may impose an upper bound on the total number of weekly tours. These added requirements can be handled by using auxiliary on-off decision variables in the model and make the job of obtaining a schedule hard. Therefore, the emphasis in this book is on obtaining a near optimal schedule rather than a globally optimal one which may take prohibitively long to yield a solution, sometimes with an unacceptable transient service level performance. A moderate amount of inefficiency introduced in a near optimal solution sometimes provides a beneficial agents buffer that dampens the swings in the transient service level from falling too far below a specified target in a system in which the call arrivals exhibit a stochastic behavior. A side effect of lower efficiency is increased cost.

This book provides code snippets that explain the application and generation of Dantzig’s model for weekly scheduling of agents in a call center and its solution using APIs (application programming interfaces) exposed by the five solvers. The authors exclusively use Microsoft’s C#.NET language for all illustration and explain how to P/Invoke (Platform Invoke) methods in the solver modules written in a non-managed code environment. The authors have chosen Microsoft’s VSTO to develop an Excel 2007 front end to provide input parameters, conduct a call center scheduling scenario using C# components executing the solver modules in a background thread, and display the schedule as tables and charts. This familiar environment, available on most desktops, offers capabilities beyond a browser needed by today’s knowledge worker. However, the components developed can be reused to develop Windows, web, console, or service oriented applications. The approach is to provide the snippets in a step by step manner through the various phases of Dantzig’s model generation and solution leading up to an end to end code for a complete console and Excel 2007 implementations.

The framework for a stochastic tool using Monte Carlo methods to simulate the service level performance of a weekly schedule for a given call load is also illustrated. The subject of stochastic simulation is widely available in the literature and will not be covered. However, the results of simulating the generated schedule to estimate transient service levels have been discussed quite extensively. Generating a weekly schedule is a deterministic activity, i.e. the schedule dictates the exact number of workers needed in each tour along with relief breaks. Once the schedule is in place, the random nature of arriving call load handled by the workers in the schedule can produce varying degrees of service levels with an expected value in each planning period. Hence, it may be an arduous task to micro manage the deterministic placement of shifts and relief breaks and find a globally optimal schedule in the context of a stochastic call arrival activity. So, the emphasis in this book is on generating a near optimal solution of Dantzig’s model in a relatively short

period of time, estimating its service level performance against the projected call load using a stochastic simulator, and regenerating the schedule, if necessary, by tuning controllable parameters so that a specified target level is met within a given tolerance. In our opinion, understanding a schedule's service level is a very important element of its overall characteristics. Several scenarios to illustrate the effect of scheduling parameters on cost, performance, and resource requirements will demonstrate the effectiveness of the model.

Adjusting the requirement vector in Dantzig's model to account for period to period dependence of call load using stochastic simulation is illustrated. The compensated requirement attenuates swings in transient service level and wait time to a greater degree compared to Stationary Independent Period by Period (SIPP) [33, 34, 35] or Erlang-C requirement. An initial attempt at an alternative formulation of Dantzig's model in the frequency domain is provided. This formulation using Fourier transform of both the requirement and tour vectors to retain only components in the low frequency range considerably reduces the model size compared to its traditional time domain counterpart. A schedule generated using this approach is provided and it compares well with that obtained using its time domain equivalent.

This practical guide is intended for technically advanced call center analysts who constantly strive to find ways to improve the performance of their centers. Knowledge of call center terminology and a working knowledge in Microsoft Excel 2007, C#, Visual Studio 2005, object oriented design, and basic linear and complex number algebra is assumed. Knowledge of Dantzig's simplex algorithm, though useful, is not required. This is implicit in the solver modules. This book's emphasis is on understanding and formulating Dantzig's model and generating schedules quickly. For introductory purposes, the solution of the model can be treated as a black box in which the solver libraries encapsulate all the functionalities needed to generate a schedule. This book facilitates the basic mechanics involved in 1) coding necessary to generate Dantzig's model matrices A , B , and C , 2) invoking the appropriate APIs in the modules that transfer the inputs to the solver engine, 3) generating a near optimal solution, 4) estimating transient service level performance of the schedule using a stochastic simulator, 5) tuning adjustable parameters, if necessary, to regenerate a schedule that meets a target level and 6) presenting the results. Once these basics are understood, the user can experiment with advanced features of the modules to optimize the solution process. *Code snippets in this book for a sample case may serve as a guide for an analyst to expand and build his/her own analysis tool. They were developed and tested using Visual Studio 2005¹ on a Wintel desktop and laptop running Windows XP Professional or Vista ultimate. The reader is free to copy and modify the code as he/she sees fit and is responsible for installing all prerequisites. The*

¹ Visual Studio 2005, Excel 2007, C#.NET, VB.NET, LINQ, VSTO, Windows XP Professional, and Vista Ultimate are products of Microsoft Corporation, Redmond, Washington.

authors provide no warranty of any kind. It is the sole responsibility of the user for the code's behavior in his/her machine and adherence to software licensing terms of the solver modules. The authors try to be concise in the presentation of the problem formulation and implementation. An in-depth view of the model, algorithm, and APIs beyond what is covered here can be obtained from the references cited. We welcome comments.

Manjini Saminathan, Bahiru Kassahun and Arun Singh
E-mail: author@sittananda.com

November 2008

To our spouses and children

Table of Contents

1	Introduction	2
1.1	Need for an analysis tool	3
1.2	Organization of this book	4
2	Application of Dantzig’s workforce scheduling model for call centers	7
2.1	Dantzig’s model	8
2.2	Model generation – The A matrix	10
2.2.1	Code Snippet 1 – Matrix A generation for example in Table 1	12
2.2.2	Dantzig matrix A (24×24) generated by code Snippet 1	14
2.2.3	Running the code snippets in Visual Studio 2005	15
2.2.4	A note on generating A matrix for multiple shift types	15
2.2.5	Dantzig matrix A for weekly tour scheduling	16
2.2.6	Code Snippet 2 – Generating A using tour basis vector in Table 2	17
2.2.7	Validity of Dantzig Matrix A	19
2.2.8	Code Snippet 3 – Checking validity of A	19
2.3	Model generation - The objective function (the C vector)	21
2.3.1	Minimize total head count (HC)	21
2.3.2	Minimize total relief hours (RH)	22
2.3.3	Minimize total work hours (WH)	22
2.3.4	Minimize total idle (slack) hours (IH)	22
2.3.5	Minimization criteria – Which one to choose?	23
2.3.6	Code Snippet 4 – Objective function weights	23
2.3.7	Output generated by snippet 4	26
2.4	Model generation – Requirement (The B vector)	26
2.4.1	Call volume, service time, and deterministic number of agents	27
2.4.2	Stochastic number of agents – Service level and Erlang-C Formula	28
2.4.3	Code snippet5 – Minimum requirement - Deterministic and Erlang-C	32
2.4.4	Output generated by code snippet 5	36
2.4.5	Difference in requirement – (Erlang-C - Deterministic)	37
2.5	Model generation - Measures	38

2.5.1	Schedule measures	38
2.5.2	Graphical representation of Dantzig's model	38
2.5.3	Summary	40
3	Model Solution – Generating the schedule	41
3.1	APIs exposed by the solver modules	43
3.1.1	Output fragment from dumpbin.exe for glpk.dll	44
3.1.2	Output fragment from dumpbin.exe for lpsolve55.dll	45
3.1.3	Accessing the APIs using P/Invoke	46
3.1.4	Skeletal code fragment for P/Invoke	47
3.2	Scheduling using GNU Linear Programming Kit (GLPK)	48
3.2.1	Minimal set of GLPK methods to solve Dantzig's model	49
3.2.2	A brief explanation of the GLPK C# solver component	50
3.2.3	Capturing messages generated by glpk.dll using C# events	51
3.2.4	Multiple tour types with different shift lengths	51
3.2.5	Code fragment that returns multiple basis tour vectors for multiple types	52
3.2.6	Obtaining a schedule with shifts starting at different time boundaries	53
3.2.7	Imposing additional constraints on Dantzig's model	54
3.2.8	Type 1 - Lower bound on the ratio of full time to part time workers	55
3.2.9	Code fragment that generates Type 1 constraints in GLPK solver	55
3.2.10	Type 2 – Lower and upper bounds on number of workers in tours	56
3.2.11	Code fragment in GLPK for Type 2 constraints	57
3.2.12	Type 3 – Upper bound on the number of tours in a schedule	58
3.2.13	Code fragment to generate Type 3 constraints	59
3.2.14	Fragment of model generated including Type 1,2 and 3 constraints	59
3.2.15	Dantzig's workforce scheduling model - Constraints summary	61
3.2.16	Skeletal framework of code snippet 6	61
3.2.17	Description of componets used in GLPK console application	62
3.2.18	Class diagram of GLPK console implementation	63
3.2.19	Code snippet 6 – Scheduling using GLPK solver module	64
3.2.20	Tour scheduling for centers with limited hours of operation	82
3.2.21	Executing code snippet 6	82
3.2.22	Weekly schedule generated by snippet 6	82

3.2.23	Weekly schedule generated by GLPK – Summary	86
3.2.24	Pictorial representation (Gantt View) of the schedule (snippet 6)	87
3.3	Scheduling using lp_solve	88
3.3.1	Minimal set of lp_solve methods to solve Dantzig’s model	89
3.3.2	A brief explanation of the lp_solve C# solver component	90
3.3.3	Capturing messages generated by lpsolve55.dll using C# events	91
3.3.4	Allowing shifts to start at different time boundaries	92
3.3.5	Type 1 constraint – Minimum of 50% full time workers in schedule	92
3.3.6	Type 2 constraint – Lower and upper bound on workers in a tour	92
3.3.7	Type 3 constraint – Upper bound on the number of tours in a schedule	93
3.3.8	Class diagram of lp_solve	94
3.3.9	Code snippet – Console application using lp_solve	95
3.3.10	Output generated by lp_solve code snippet	106
3.3.11	Schedule generated by lp_solve	107
3.3.12	Weekly schedule generated by lp_solve – Summary	111
3.3.13	Gantt view of the schedule generated by lp_solve	112
3.4	Scheduling using NEOS solvers (Feaspump and SCIP)	113
3.4.1	Communicating with NEOS XML-RPC Server	114
3.4.2	Steps involved in submitting a job to NEOS	116
3.4.3	NEOS Feaspump solver	117
3.4.4	FEASPUMP XML template to submit Dantzig’s model to NEOS	117
3.4.5	Code fragment to populate the FEASPUMP XML template	118
3.4.6	NEOS SCIP Solver	119
3.4.7	SCIP XML template to submit Dantzig’s model to NEOS	119
3.4.8	Code fragment to populate SCIP XML template	120
3.4.9	Parsing the output from NEOS Feaspump and SCIP	121
3.4.10	A skeletal framework of NEOS console application	123
3.4.11	Class diagram of ConsoleNeos for NEOS Feaspump and SCIP solver	126
3.4.12	Code snippet – Scheduling using NEOS Feaspump and SCIP	127
3.4.13	Output generated by NEOS Feaspump algorithm	146
3.4.14	Output generated by NEOS SCIP	153
3.4.15	Weekly schedule generated by NEOS Feaspump and SCIP – Summary	161

3.4.16	Gantt view of the schedule generated NEOS FeasPump.....	162
3.4.17	Gantt view of the schedule generated NEOS SCIP.....	163
3.5	Scheduling using SCIP on a local machine	164
3.5.1	Code fragment to spawn a SCIP process on a local machine	164
3.5.2	A note on SCIP batch file, settings and scheduling parametrs	166
3.5.3	Class diagram for SCIP	167
3.5.4	Code snippet(Console application) – Scheduling using standalone SCIP	168
3.5.5	Output generated by SCIP	178
3.5.6	Weekly schedule generated by SCIP.....	184
3.5.7	Weekly schedule generated by SCIP – Summary.....	186
3.5.8	Percent gap and execution time – SCIP.....	186
3.5.9	Gantt View of the schedule using SCIP (including Type 1,2 and 3)	187
3.6	Scheduling using COIN-OR branch and cut (Cbc) solver.....	188
3.6.1	cbcSolve.exe arguments that control solver execution	189
3.6.2	Class diagram of Cbc console application.....	190
3.6.3	Code snippet (Console application) scheduling using Cbc	190
3.6.4	Output generated by Cbc	201
3.6.5	Weekly schedule generated by Cbc.....	202
3.6.6	Gantt view of the schedule generated by Cbc.....	206
3.6.7	Weekly schedule generated by COIN-OR Cbc – Summary.....	206
4	Monte Carlo simulation of schedule service level.....	207
4.1	Erlang-C simulator	208
4.1.1	Erlang-C Simulator schematic	208
4.1.2	Simulation assumptions	209
4.1.3	Generating incoming call arrivals (Poisson Process)	209
4.1.4	Generating service times (Exponential distribution).....	209
4.1.5	Code fragment – Generating inter arrival and service times	210
4.1.6	Validating the simulation model	210
4.1.7	Service level results for Erlang-C simulator - validation	212
4.2	Schedule service level simulator	214
4.2.1	Weekly schedule to be simulated.....	215
4.2.2	Schematic of schedule service level simulator	216

4.2.3	Schedule service level simulator	217
4.2.4	Summary of simulation runs (1 thru 4)	218
4.2.5	Simulation run 5 – Service level and wait time	219
4.2.6	Performance measures	220
5	Excel 2007 Implementation of Dantzig’s model	221
5.1	Excel 2007 interface	222
5.1.1	Ribbon, tabs and groups in Excel 2007	223
5.1.2	Ribbon1.xml file to configure a new tab “Dantzig 2007”	223
5.1.3	Custom task pane	223
5.1.4	“Input Parameters” task pane added to Excel 2007	224
5.1.5	Creating an Add-In and Ribbon support in Excel 2007	225
5.1.6	Skeletal framework of user control inputs.cs	226
5.1.7	Displaying messages and charts in an Excel sheet.....	229
5.1.8	Snapshots of a scheduling run in Excel 2007	231
5.1.9	Deploying Excel2007 and VSTO solutions.....	234
5.1.10	Code snippet (Excel Add-In - ThisAddIn.cs)	235
5.1.11	Code snippet – Ribbon1.cs.....	236
5.1.12	Code snippet – User Control inputs.cs	238
5.1.13	Code snippet – DrawChart.cs.....	249
6	Parametric Study.....	258
6.1	Parameters, measures, and performance.....	258
6.1.1	Effect of objective function	261
6.1.2	Effect of shift increment	263
6.1.3	Effect of understaffing.....	265
6.1.4	Effect of increasing call load	267
6.1.5	Effect of solvers	269
6.1.6	Effect of deterministic versus Erlang-C requirement.....	271
6.1.7	Effect of type 1, 2, and 3 constraints.....	272
6.1.8	Sub-optimality, solver, efficiency, solution time, and service level	277
6.1.9	NEOS Feasibility Pump on schedule quality.....	280
6.1.10	Cost and performance.....	282
7	Adjusting Requirement for Period Dependence.....	283

7.1	Actual versus projected call load	285
7.1.1	Estimating actual call load using Monte Carlo simulation.....	286
7.1.2	Code fragment to estimate actual versus offered load.....	288
7.1.3	Filtering noisy Actual/Offered data using Fourier techniques.....	289
7.1.4	Adjustments to SIPP requirement B	290
7.1.5	Transient service level after adjustment for SIPP	291
7.1.6	SIPP Compensation summary - Perfect schedule	293
7.1.7	SIPP Compensation summary – A real schedule.....	294
8	Frequency domain solution of Dantzig’s scheduling model.....	296
8.1	Time domain method.....	296
8.2	Frequency domain formulation and solution	298
8.2.1	Frequency spectrum of requirement vector B	298
8.2.2	Requirement – Bandwidth limited reconstruction	299
8.2.3	Frequency spectrum of 5x9 hour tour with relief breaks.....	300
8.2.4	Tours – Bandwidth limited reconstruction	301
8.2.5	Formulation of Dantzig’s model in the frequency domain.....	302
8.2.6	Code fragment to generate the matrices in (31) using GLPK.....	305
8.2.7	A fragment of the frequency domain model generated by GLPK	307
8.2.8	Schedule generated using frequency domain model.....	309
8.2.9	Gantt view of the schedule - Frequency domain model.....	311
8.2.10	Service level and wait time – Frequency domain model.....	312
8.3	Formulation of Dantzig’s model using convolution.....	313
8.3.1	FFT Convolution of two discrete time signals.....	313
8.3.2	Formulation of Dantzig’s model using convolution	313
8.3.3	Convolution model parameters	315
8.3.4	Code fragment to generate the matrices in (37) using GLPK.....	315
8.3.5	A fragment of Dantzig’s model generated using FFT convolution	317
8.3.6	Schedule generated using convolution model.....	319
9	A summary evaluation of solvers for time domain models	322
9.1	Evaluation Methodology.....	323
9.1.1	GLPK Solver.....	324
9.1.2	NEOS SCIP.....	325

9.1.3	lp_solve.....	326
9.1.4	NEOS Feasibility pump.....	326
9.1.5	Standalone SCIP/SoPlex.....	327
9.1.6	COIN-OR Cbc solver.....	327
10	Summary.....	328
11	References.....	330
12	Appendix 1.....	334
12.1	Weekly schedule and Service Level for a sample case.....	334
12.1.1	Schedule Measures Summary:.....	334
12.1.2	Weekly call volume and SIPP Requirement.....	335
12.1.3	Gantt view of the schedule:.....	335
12.1.4	Idle hours (Bread Crumb Plot).....	336
12.1.5	Simulated service level and wait time (Runs 1 thru 2).....	336
12.1.6	Simulated service level and wait time (Runs 3 thru 5).....	337
12.1.7	Weekly Schedule.....	338
12.2	Solvers, Servers, Source, and Licensing Terms.....	345
12.3	Fourier components for the requirement vector B	346
12.4	Fourier components for tour vectors in tour matrix A	348
13	Appendix 2.....	355
13.1	Schedule service level simulator – Class Diagram.....	355
13.2	Monte Carlo Simulation – Code.....	356
13.2.1	MonteCarlo.cs.....	356
13.2.2	CCSimulator.cs.....	358
13.2.3	Simulator.cs.....	362
13.2.4	Call.cs.....	368
13.2.5	Operator.cs.....	370
13.2.6	ArrivalAndServiceTimeGenerator.cs.....	370
13.2.7	CCRunsSummary.cs.....	371
13.2.8	CallStatistics.cs.....	379
13.2.9	CallWithLowestCompletionTime.cs.....	382
13.2.10	SimulationController.cs.....	383
13.2.11	SortAscendingBusyCalls.cs.....	383

13.2.12	SortAscendingIncomingCalls.cs.....	384
13.3	Fourier transform component	385
13.3.1	Fourier transform Code using FFTW library in reference [36]	385
13.3.2	P/Invoke signatures to access methods in FFTW [38]	389
13.4	SIPP adjustment to Erlang-C requirement vector B	390
13.4.1	SIPP compensation code.....	390
13.5	Component to generate Gantt view of the schedule	391
13.5.1	MatrixChart.cs.....	391
13.6	Nomenclature.....	396

A practical guide for call center analysts to implement Dantzig's seminal workforce scheduling model to generate, evaluate, and optimize service level performance of weekly agent schedules using Excel 2007, C#.NET components and publically available solvers, algorithms, and hardware. Book features:

- Introduction to Dantzig's seminal workforce scheduling model: Minimize $C(\text{transpose})X$ such that $AX \geq B$, $X \geq 0$. Formulation of tour (A), requirement (B), and objective (C) matrices for weekly tour scheduling of agents
- Model solution using public solvers (GLPK, Ip_solve, and NEOS Feasibility Pump and SCIP) and local/public hardware
- Scheduling with upper bound on total number of weekly tours, each tour having lower and upper bounds on number of agents. C# code snippets to create console and Excel 2007/VSTO applications
- Monte Carlo stochastic simulation to estimate schedule service level and ASA. Effects of controllable parameters on schedule quality
- Near optimal scheduling to meet a target service level for a given call load. Adjust SIPP (Erlang-C) requirement for period dependence to dampen "roller coaster" or "bull whip" effect in service level performance
- A novel frequency domain reformulation of Dantzig's time domain model to reduce size by 75% by retaining only low frequency Fourier components of the requirement (B) and tour (A) vectors

Manjini Saminathan is a software architect and consultant who currently develops engineering applications using the wealth of publically available models, algorithms and software libraries. He has more than two decades of experience in Bell Labs and Lucent in modeling, simulation and optimization of IC manufacturing processes and system assembly operations. He holds a PhD in Chemical Engineering from WPI, Worcester, Massachusetts.

Bahiru Kassahun's expertise is in modeling and optimizing thermal and polymeric processes in semiconductor and optical fiber manufacturing. His current focus is in Finite Element modeling of flow processes. He has more than two decades of experience in Bell Labs and Lucent and holds a PhD in Engineering Mechanics from ASU, Tempe, Arizona.

Arun K. Singh has a wide variety of experience in system engineering, combustion and engineering analysis. He has over 20 years of experience in Bell Labs in design and specification of high-speed optical networks. His focus is in queuing theory and discrete event simulation. He holds a PhD in Mechanical and Aerospace engineering from Rutgers University, New Brunswick, New Jersey.

