Applying an Alternative Heuristic Algorithm to Gupta Method for Sequencing Lean Software Processes

Eldon Caldwell
University of Costa Rica

Oscar Gamboa
Technological Institute of Costa Rica

Abstract

Gupta algorithm (1972) was proposed in order to reduce cycle times, with very good performance results and it has been successfully used as a sequencing criterion in critical activities in software development processes. An alternative to this algorithm is developed and presented in 2009 by Caldwell (2009) using heuristic rules when equal selecting factors occurs. This document presents the results of a Quasi-experimental method in order to explain how to apply Caldwell-T algorithm improving software development projects, when it is applied to critical chains and feeding buffers and signal calculations. In these environments, the shared resources affect not only critical path but also tasks with time buffers and multiple nodes of dependency, which requires the search for reducing makespan in order to accelerate the affected routes. The new algorithm has been validated through a simulation model using a randomized procedure resulting in 84% performance effectiveness, referred to makespan, over Gupta solution after 2000 iterations. However, we conclude that new algorithm can obtain results less effective against Gupta for certain random configurations in approximately 18% of the cases; therefore it is better to analyze the multi-project environment configuration with both algorithms, rather than making them compete knowing that both algorithms have the same degree of simplicity in calculation and application by automatic ways.

Keywords
Critical Chain, Gupta Algorithm, Lean Software Development, sequencing, agile methods.

1. Introduction

The software application development has been extensively studied from various perspectives, such as project management, resource management practices, the use of building code standards, good practice teamwork, etc. Development methodologies have evolved from the centralized architecture to focusing on the development of functionalities rather than systems.

Within the current study and optimization of development methodologies, as opposed to the traditional methodology or waterfall, agile approach, particularly lean software development, has become important and followers.

At the beginning of the nineties, Womack, Roos and Jones (1991) in the United States, published a book that changed the way we understand the process flow. This book titled: The Machine That Changed the World (1991) and in it, authors planted the basis of what has been popularized as lean (Lean Manufacturing). This concept caused such a stir in academic and business circles, which forced two of the authors (James Womack and Daniel Jones) to write a more generic, close to a school of thought called optimization and "lean thinking".

The lean thinking has been strengthened with the passing of the years, enhanced by the application with specific techniques such as Six Sigma, leading some diverse derivatives: Lean Six Sigma, Lean Service Management, Lean Accounting and Lean Supply Chain Management. Given these developments, it was foreseeable that software development eventually also has to accept this line of thought as one of the alternatives (George M., 2003).

One of the problems that require innovative engineering solutions is the sequencing of tasks in an environment with multiple share resources and multiple projects. Specially, when these critical resources are expensive and affect the profitability of the project. To solve this problem, researchers have explored the application of sequencing rules and heuristic algorithms that seek downtime reduction and makespan (Modrak V. & Pandian R., 2010). This point is where we started our motivation for this research, aware that the results could impact very positively to the software industry and other sectors that focus their operations on the implementation of projects.

Here, we present an alternative way to solve the problem of ordering the tasks in an optimal sequence to allow greater use of those share resources in critical chain project management of software development and also the complexity of tasks concerning multiple projects.

2. Literature Review

Mary and Tom Poppendieck (2003) performed slender thought that translation of the software development process, applying seven categories of waste, typically presented in them. They took Taichii Ohnno´s proposal (1976) referred of seven basic categories of waste in the process flow and delimited version for the case of software development projects: work partially performed, unnecessary work, unnecessary functionality, frequent changes of activity (the effect multitasking), waiting, motion, defects and under-utilization of human talent (Modrak & Pandian, 2010).

In order to attack this waste categories, the researchers proposed what they considered medicines or early Lean Software Development principles: eliminate waste, build quality, create knowledge, defer commitment, deliver fast, respect for people and optimizing the whole (Poppendieck & Poppendieck, 2006).

Based on Bain (2008), eliminating waste is the primary guideline for the Lean practitioner. Waste, for example, is code that is more complex than it needs to be. Waste occurs when defects are created. Waste is non-value-added effort required to create an application. Wherever there is waste, the Lean practitioner looks to the system to see how to eliminate it because it is likely that an error will continue to repeat itself, in one form or another, until we fix the system that contributed to it (Bain R., 2008).

Deferring commitment means to make decisions at the right time, at the “last responsible moment”: don’t make decisions too early when you don´t have all the information, and don’t make them too late, when you risk incurring higher costs. Therefore, deferring commitment is a pro-active way to plan the process so that we either don’t work on something until we need to or we set it up so that we can make decisions that can be reversed later when we get more information. This principle can be used to guide requirements, analysis, and system design and programming (Poppendieck & Poppendieck, 2006).

In this scientific paper, we worked on eliminating waste caused by unnecessary waiting time on activities that are within the critical paths of multiple projects using shared resources, such as developers. Moreover, by appropriate sequencing of tasks delayed, decision making process, either before or after required, also can be improved, ie when project managers typically reallocate resources without visibility of the overall effect of critical path delays over multiple projects. We started with the assumption that resource allocation and scheduling in multi-project environments with shared resources, can make a big improvement in the performance and profitability of software development projects (Modrak & Pandian, 2010).

More specifically, the delays in software development projects introduce additional costs associated with the waste of resources that wait as well as quality hidden costs and underutilization of qualified personnel. For this reason, the sequencing of activities when path crashes occur in environments of multi-project critical chains, introduce a complex context where the application of algorithms can be an effective solution (Bain, R., 2008).

An algorithm is like a logical recipe for solving a model. A case is a specific data set for that model. Exact algorithms (optimal) provide an optimal solution for each case of the problem and, on the other hand, heuristic algorithms provide solutions that are expected to be optimal or near-optimal in all cases (Modrak & Pandian, 2010). The reason why optimal approaches are not used to solve task sequencing or work orders is because in many programming models, the only known exact solutions are based on enumeration, such as dynamic

programming. In practical cases, the combinatorial nature of the production order scheduling problem makes the approach very complex in a computational perspective (Artigues and Buscaylet, 2003).

Heuristic algorithms are judged on their effectiveness, in other words, how the solution meets the objectives, analyzing the efforts made to obtain it. This effectiveness can be validated theoretically or empirically. Empirical evidence is the most used and helps to generate and solve many practical problems in different contexts. For the cases in which it is not possible to obtain an optimal solution in order to compare it with the solutions generated by a heuristic algorithm, the solution is compared to a mathematical bound on the optimal result. If there is a small difference, the effectiveness of the heuristic is good and if the algorithm has a good performance in the test cases, it is the assumption that performs well in other cases (Zhang Shuli, 2013). This may not be true for any case that differs from testing scenario and the solution should be compared with the current heuristic solution (Toro E., Granada M., Romero R., 2005). For all heuristic algorithms developed and published, there is no guarantee that the solutions are good or not even that can be used in certain practical situations.

Over the past 60 years there have been several heuristic algorithms to optimize the solution to the above problem:

1. Johnson Algorithm (1954-1957) : 2XN, 3XN
2. Campbell Algorithm (1966-1970): MXN, N-1 sequences
3. Gupta Algorithm (1972): MXN, two phases
4. Gupta Validation (TSung-Chyan Lai, 1996)
5. Genetic Algorithm (Chu Beasley,1997)
6. Artificial Intelligence Alternative Algorithms (Zimmermann, A., Dalkowski, K., Hommel, G., 1996)
7. Tabu Search Method by Artigues C. And Buscaylet F. (2003): MXN
8. Beasley Algorithm validation (Toro E., Granada M., Romero R., 2005)
9. Caldwell-T Algorithm (Caldwell E., 2009): MXN
10. Modrák V. & Pandian R. Algorithm (Modrák V. & Pandian R., 2010): MXN
11. Sequence List Algorithm by Shuli Zhang, 2013

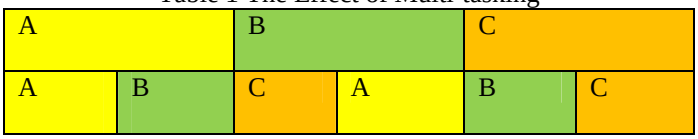
The two most robust algorithms to solve this problem are proposed by Campbell (1966), Campbell, et al (1970), Gupta (1972), Modrak and Pandian (2010) and Caldwell E. (2009); all demonstrate mathematically that the difference is negligible in the final results of seniority, with ties in ranking criteria.

3. An application of Caldwell-T Algorithm according with Lean Software Development context

Software development projects have a very special dynamic: they are, in themselves, a project of knowledge generation and on the other hand, team learning. The common practice is that many developers perform many activities, several of which are highly complex and require inspiration and talent together.

The complexity of a software development project is manifold: a code constructs logical level alignment with the reporting requirements and the needs of users, shares, varying periods of concentration and inspiration programmers, changes in the demands of the users, etc. A traditional approach to work, in an environment controlled by the uncertainty, is, in many cases, the work in a multitasking way. The general approach to multitasking is relative ineffective, as noted in Table 1.

Table 1 The Effect of Multi-tasking



As seen in the example of Table 1, the ABC task, dedicated mode sequenced, are completed in less time than the same job with multitasking sequence. In the case of the task C, there is no gain at the time of completion.

In common to specific projects related with software development there are activities implemented by shared resources. Since, this implies the need to determine an optimal sequence.

Frequently we can find the existence of overlapping shared resources across multiple projects, then as a solution, one of the most used rules of overlapping problem is First Incoming Outgoing First (FIFO-First In First Out).

It is quite true that in an infinite number of iterations, the randomness does not generate a significant asymptotic limit to draws on the criteria of the two most commonly used algorithms (Campbell and Gupta). But behind the scenes, there are differences that may be important to a particular project.

Suppose we have a number of activities concerning various projects, and that must be executed for shared resources, as expressed in Table 2.

Table 2 Sequence Problem

	Tasks				
Resources	A	B	C	D	E
1	66	27	80	72	64
2	28	60	14	12	92
3	38	22	26	28	74
4	2	18	100	78	6
5	76	15	100	94	81
Gupta Factor	-0,025	0,0303	-0,025	-0,025	-0,0125
SUM	210	142	320	284	317

Gupta algorithm is based on a discriminating factor and is calculated as follows:

Ki = 1 / (Min (ti1+ti2+.....+tin-1+tin))

Multiply k:

Times 1 if ti1 >= tin
Times –1 if ti1 < tin

ti is the estimated time for each activity and n is the number of activities.

After calculating Ki, factors are ordered from lowest to highest and so the sequence is obtained.

In this case, the recommended sequence by Gupta is:

CDAEB

With this sequence, the total time for completion of projects is 586 minutes. As noted, there is a tie for the activity factor A and C.

However, considering a tiebreaker in this case, we can find another solution algorithm through Caldwell-T.

This algorithm is written as follows (Caldwell E., 2009):

1-First we calculate a ranking factor Factor B

$Bi = 1 / (\min ((T_{i,1}+T_{i,2}),.....(T_{i,n-1}+ T_{i,n}))$

$T_{i,n}$ is the processing time of order i in the operation n.

Ti is the processing time of the activities and n is the number of activities.

2- Multiply k:

Times 1 if ti1 >= tin
Times –1 if ti1 < tin

3-After assigning the sign of Bi factor, the best sequence is ordered from lowest to highest results.

4- In case of a tie between Bi factor values, select the job with the lowest processing time. If still tied, selects the job with the lowest cycle time between the first two operations and so on until the operation M.

In the case above, we found that this new recommended sequence is ADCEB algorithm, resulting in a total time of completion of projects (subgroups could be full use case) of 554 minutes against 586 minutes the algorithm recommended Gupta. Moreover, when compared with the algorithm Campbell (1967), this algorithm is also robust because the sequence Campbell DCEAB sheds with a time of 564 minutes.

4. Caldwell-t Algorithm Application: A Quasi-Experimental Validation using a Simulation Model

First step taken in this research was to generate 40 random iterations provoke draws and discrimination factors, taking a random sample of data on the implementation of software development projects that is based on Lean Software Development. This sample was selected from a database belonging to an IT services company located in Costa Rica with over 15 years of experience. Figure 1 shows the results.

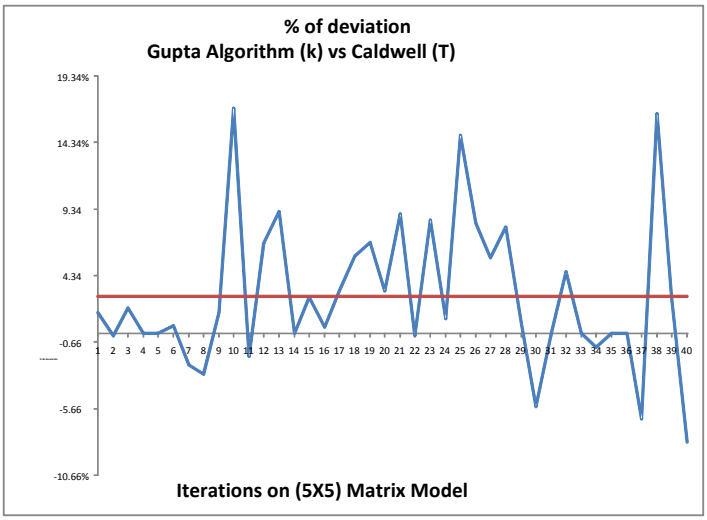


Figure 1: Forty random iterations of the new algorithm.

As noted, there are high variations in the results, but it is remarkable that when Caldwell-T yields lower make span time against Gupta algorithm does so much. But equally, though less likely to occur when the Caldwell-T generates higher against Gupta times, it does so much.

This is seen in Table 3, which shows that the new algorithm provides an equal or better solution than Gupta in 81% of cases and with a negligible variation range of about 17%.

Table 3: Effectiveness of Caldwell-t Algorithm

	Caldwell-t over Gupta
Range	0%
	16,91
Effectiveness	81%

If the procedure is repeated, but this time for a very large number of iterations, we can find a more accurate result.

The result for 2000 iterations generated random standardized way is shown in Table 4.

Table 4: Effectiveness for 2000 random iterations.

	Caldwell-t over Gupta
Range	0%
	18,62
Effectiveness	84%

As can be seen, Caldwell-T provides better results than or equal to Gupta algorithm by nearly 85% with a range of variation of 18.62%, which demonstrates a significant improvement.
In this research, we conducted a statistical reliability test, resulting in a 96% confidence and an error of 3.4%.

5. Conclusion

Caldwell-t algorithm is shown to be more robust against Gupta algorithm to reduce make-span in environments characterized by lean software development practices. Appreciably, Caldwell-t method could reduce waste in sequencing multi-projects shares models and multitasking environments. Clearly, the Caldwell-t could become a tool for optimizing the execution of multiple projects to run faster small routines programming cycles use cases. It also can be used to effectively solve, potential activity shocks that may occur on shares, compared to the traditional methodology that encourages multitasking operation. However, we conclude that new algorithm can obtain results less effective against Gupta for certain random configurations in approximately 18% of the cases; therefore it is better to analyze the multi-project environment configuration with both algorithms, rather than making them compete knowing that both algorithms have the same degree of simplicity in calculation and application by automatic ways. Moreover, from an economic standpoint, the software development projects are typically expensive from the point of view of resources invested and hidden costs associated with delays in critical activities. So saving time, even in only 18% of cases, could make a big difference in the total cost of development.

6. Future Research

This research shows that the development of rules and algorithms playoff could generate benefits in reducing development times and IT projects could facilitate the adoption of lean practices. As a future line of research, there is the exploration of these algorithms for different configurations of development projects with multi-paths and specific variability in the time of completion of activities.

7. References

Artigues C. And Buscaylet F., A fast tabu search method for the job-shop problem with sequence-dependent set up times. *Proceedings of the Metaheuristic International Conference MIC 2003*, Kyoto, Japan, 2003.

Bain R., Emergent Design: The Evolutionary Nature of Professional Software Development. Addison-Wesley, Boston, USA, 2008.

Caldwell E., Lean Manufacturing: Fundamentals and Technics for Cycle Times Reducing. UACA-Kaikaku Institute Publishers, San Jose, Costa Rica, 2009.

Campbell, H., Dudek R.A. & Smith, M.L., A Heuristic Algorithm for the n Job, m Machine Sequencing Problem, Management Science, 16, B630-B637, USA, 1970.
Campbell, H., A heuristic technique for near optimal production schedule, Ph.D. Thesis. Texas Technological College, USA, 1966.
Tsung-Chyan Lai, "A note on heuristics of flow-shop scheduling", Operations Research, Vol. 44, No. 4 (pp. 648-652), 1996.
Poppendieck Mary & Poppendieck Tom, Lean Software Development: An Agile Toolkit, Addison Wesley, Boston, USA, 2003.
Poppendieck Mary & Poppendieck Tom, Implementing Lean Software Development: From Concept to Cash, Addison Wesley, Boston, USA, 2006.
George Michael, 2003, Lean Six Sigma for Services McGraw-Hill, Washington, USA, 2003.
Gupta, J.N.D., 1972.Heuristic Algorithms for Multistage Flow Shop Problem, AIIE Transactions, 4, 11-18, NY, USA.
Hamilton Bruce, 2007. Toast Kaizen. Greater Boston Manufacturing Partnership, NY, USA.
Modrak V. and Pandian S., 2010. Flow Shop Scheduling Algorithm to minimize completion time for n-Jobs m-Machines Problem, Technical Gazette, Vol. 17, No. 4, 273-278, Italy.
Ohno Taichii, 1976. Toyota Production System. Productivity Press., NY, USA
Shuli Zhang, 2013. A Sequence List Algorithm for the Job Shop Scheduling Problem, The Open Electrial & Electronic Engineering Journal, Vol 16, No 2, 55-61, USA.
Toro E., Granada M., Romero R. "Algoritmo Genético aplicado a la solución del problema de Asignación Generalizada". Revista Técnica UTP. Año 8, No 16, Panamá, 2005.
Womack James and Daniel Jones, 2003, Lean Thinking Simon and Schuster, NY, USA.
Womack James, R. Roos & D. Jones, 1991, The Machine that changed the world. Simon and Schuster, NY, USA.
Zimmermann, A., Dalkowski, K., Hommel, G., 1996. A Case Study In Modeling And Performance Evaluation Of Manufacturing Systems Using Colored Petri Nets". 8th European Simulation Symposium (ESS '96), France.

Biography

Eldon Glen Caldwell Marín, full professor (Cathedraticus), University of Costa Rica with over 20 years of teaching and research experience. Bachelor and Master of Industrial Engineering at University of Costa Rica, Dr. Caldwell earned a Masters degree in Operations Engineering at ITESM, Mexico, Master of Financial Analysis and M.Sc. in Marketing of Services, Inter-american University of Puerto Rico, Master of Health Services Management, UNED, Costa Rica and finally Ph.D. in Industrial Engineering at the Autonomous University of Central America-University of Nevada, Suma Cum Laude and Academic Crown Award. Currently he is a doctoral researcher in Artificial Cognition at the University of Alicante, Spain, and Academic Excellence Prized at the Ph.D. Program in Inclusive Education at University of Costa Rica. His research interests include information retrieval, machine learning, robotics and intelligent development of methodologies for implementing lean systems. His emails are eldon.caldwell@ucr.ac.cr, egcm@alu.ua.es.

Oscar Gamboa, Full Professor (Cathedraticus), Technological Institute of Costa Rica, Director of the M.Sc.Program in Manufacturing Modern Systems. He earned his Bachelor and Master of Science degree in Production Engineering at Tech Institute of Costa Rica and his experience is supported in more than 20 years as a researcher, trainer and consultant in operations planning and scheduling in factories located in North and Latin America. His email is ogamboa@itcr.ac.cr.