



Queensland University of Technology
Brisbane Australia

This is the author's version of a work that was submitted/accepted for publication in the following source:

[Agdas, Duzgun, Warne, David J.](#), Osio-Norgaard, Jorge, & Masters, Forrest J.
(2018)

Utility of Genetic Algorithms for Solving Large Scale Construction Time/Cost Trade-off problems.

Journal of Computing in Civil Engineering, 32(1), pp. 1-10.

This file was downloaded from: <https://eprints.qut.edu.au/107593/>

© © 2017 American Society of Civil Engineers

License: Creative Commons: Attribution-Noncommercial 4.0

Notice: *Changes introduced as a result of publishing processes such as copy-editing and formatting may not be reflected in this document. For a definitive version of this work, please refer to the published source:*

[https://doi.org/10.1061/\(ASCE\)CP.1943-5487.0000718](https://doi.org/10.1061/(ASCE)CP.1943-5487.0000718)

UTILITY OF GENETIC ALGORITHMS FOR SOLVING LARGE SCALE CONSTRUCTION TIME-COST TRADE-OFF PROBLEMS

Duzgun Agdas, PhD¹, David J. Warne², Jorge Osio-Norgaard³, and Forrest J. Masters, PhD⁴

¹School of Civil Engineering and Built Environment, Queensland University of Technology (QUT), 2 George Street, Brisbane, QLD 4001, Australia. Email: duzgun.agdas@qut.edu.au

²High Performance Computing and Research Support, Queensland University of Technology (QUT), 2 George Street, Brisbane, QLD 4001, Australia.

³The Department of Civil, Environmental and Architectural Engineering, University of Colorado Boulder, Boulder, CO, 80309, USA.

⁴Herbert Wertheim College of Engineering, University of Florida, Gainesville, FL 32611, USA.

ABSTRACT

The Time/Cost Trade-off (TCT) problem has long been a popular optimization question for construction engineering and management researchers. The problem manifests itself as the optimization of total costs of construction projects that consist of indirect project costs and individual activity costs. The trade-off occurs as project duration and, as a result, indirect project costs decrease with reduced individual activity duration. This reduction in individual activity duration is achieved by increasing resource allocation to individual activities, which increases their costs to completion. Historically, metaheuristic solutions have been applied to small scale problems due to computational complexities and requirements of larger networks. Findings in this article demonstrated that the metaheuristic approach is highly effective for solving large scale construction TCT problems. A custom Genetic Algorithm (GA) is developed and used to solve large benchmark networks of up to 630 variables with high levels of accuracy (<3% deviation) consistently using computational power of a personal computer in under ten minutes. The same method can also be

24 used to solve larger networks of up to 6,300 variables with reasonable accuracy ($\sim 7\%$ deviation)
25 at the expense of longer processing times. A number of simple, yet effective, techniques that
26 improve GA performance for TCT problems are demonstrated; the most effective of which is a
27 novel problem encoding, based on weighted graphs, that enables the critical path problem to be
28 partially solved for all candidate solutions *a priori*, thus significantly increasing fitness evaluation.
29 Other improvements include parallel fitness evaluations, optimal algorithm parameters, and the
30 addition of a stagnation criteria. This article also presents some guidelines of optimal algorithm
31 parameter selection through a comprehensive parameter sweep and a computational demand profile
32 analysis. Moreover, the methods proposed in this article are based on open source development
33 projects that enable scalable solutions without significant development efforts. This information
34 will be beneficial for other researchers in improving computational efficiency of their solution in
35 addressing TCT problems.

36 INTRODUCTION

37 The construction industry is characterized by non-repetitive, complex projects, with generally
38 unpredictable productivity figures. The level of complexity increases exponentially as the scope
39 and size of projects increase. For various reasons, such as meeting deadlines to avoid liquidated
40 damages, earn bonuses etc., crashing construction project schedules is common practice in the
41 construction industry. This is generally achieved through increased resource allocation to individual
42 activities to reduce their duration. In most cases, both the increase in resources and reduction in
43 duration are coupled in discrete pairs to simplify the problem (Sonmez and Bettemir 2012). The so-
44 called Time/Cost Trade-off (TCT) problem manifests itself in finding the optimum balance of total
45 project costs. That is the balance of increased activity costs and reduced indirect costs because of
46 reduced project duration—as a result of reduced activity duration (Chassiakos and Sakellariopoulos
47 2005). In its most basic form, this problem can be represented as:

$$48 \quad C_T = C_A + C_I + D - I. \quad (1)$$

49 where: C_T is total project cost, C_A is total activity cost, C_I is indirect project costs, D is
50 disincentives, and I is incentives. In reality, disincentives are more likely to be in a liquidated
51 damage format that is applied when the project duration exceeds a predetermined threshold and
52 incentives are bonuses rewarded for days saved from a specified threshold. The complication in
53 minimizing the total project cost, C_T , arises due to the large number of time-cost pairs for individual
54 activities, and the resulting network computations because of differing activity duration values. It is
55 typical for large construction projects to consist of hundreds of activities. Considering the potential
56 number of discrete time-cost pairs for these activities, combined with the logical sequencing of
57 activities, the computational demands can be extreme.

58 **Time-Cost Trade-off Problem**

59 Heuristic and metaheuristic methods have been the main tools in addressing this problem due
60 to the above mentioned computational requirements, although exact methods have also found some
61 limited interest (Kandil et al. 2010; Sonmez and Bettemir 2012; Chassiakos and Sakellaropoulos
62 2005; Boussaïd et al. 2013). Of the metaheuristics methods used in construction and in general,
63 Genetic Algorithms (GA) have been undoubtedly the most prominent method used for a multitude
64 of problems, including the TCT problem (Boussaïd et al. 2013). Inspired by the Darwinian theories
65 on biology and evolution, GA use genetic operations of cross-over, mutation and selection to find
66 near optimal solutions to otherwise complicated problems using an initial set of candidate solutions
67 (Michalewicz 1999). Possible solutions are represented as a genome upon which genetic operations
68 such as mutation and crossover can be performed, thus simulating the natural selection process.

69 Table 1 represents the state of the practice in regards to the TCT problem in the construction
70 industry. It shows the typical size of the problems solved, the methods by which they are solved,
71 performance of the methods used, and whether parallel computing was used to reduce the pro-
72 cessing time. It is clear that the majority of the articles addressed smaller networks, and detailed
73 performance benchmarks on solution accuracy and runtime performance also appear to be lacking.

74 In a broader context, the TCT problem can be considered a sub-set of the more general resource
75 constrained scheduling (RCS) problem; a heavily studied problem in operations research (Hartmann

76 and Briskorn 2010; Vanhoucke and Debels 2007). It should be noted that in construction engineer-
77 ing and management literature TCT and RSC problems appeared to have been treated as different
78 topics (Kim and Ellis Jr 2008; Zhang 2011); however, it is safe to say both belong under the same
79 class of scheduling benchmark problems. The algorithms for RCS in areas of computer science,
80 operation research and operations management can be sophisticated (Debels and Vanhoucke 2007;
81 Gaglioppa et al. 2008; He et al. 2017). Even in these disciplines, algorithm performance is often
82 only reported for relatively small networks of less than 300 nodes. Therefore, demonstration of
83 performance of large scale, realistic, TCT problems is one important contribution of this work for
84 construction engineering and management research and beyond.

85 **Issues in Solving the TCT Problem Using GA**

86 Computational complexity becomes relevant and problematic with the increased network size
87 as cost computations are simple formulas. This can explain that the majority of the earlier studies—
88 and this trend appears to continue in the more recent studies—on this subject has revolved around
89 smaller networks (Feng et al. 1997; Hegazy 1999; Chassiakos and Sakellaropoulos 2005; Long and
90 Ohsato 2009; Ghoddousi et al. 2013; Monghasemi et al. 2015; Tran et al. 2016). Works of Feng et al.
91 (1997), and Chassiakos and Sakellaropolous (2005) has addressed networks of 18 and 29 variables
92 respectively. These two articles represent some of the more popular networks that have later been
93 used in literature as key benchmark problems. With the advancements in computational efficiency
94 and better understanding of these problems, larger networks have also been analyzed (Kandil and
95 El-Rayes 2006; Kandil et al. 2010; Sonmez and Bettemir 2012). Perhaps, the most interesting
96 article on scale of problems addressed in literature has been the publication by Menesi et al. (2013),
97 in which the authors criticize the earlier literature because of the size of the networks analyzed, and
98 urge other researchers to focus on methods that can provide fast, accurate solutions to large scale
99 TCT problems. Following this criticism of (Menesi et al. 2013), this article is an attempt to improve
100 the state of GA optimization practice by taking a more holistic approach to performance analysis that
101 considers accuracy, convergence time, computational efficiency, and ease of development. Thus,
102 findings presented are highly relevant for real life scale applications; in particular, the approach to

103 parameter selection step should serve as a useful guide for practical use of GA for construction
104 TCT problems.

105 **Research Motivation and Contribution**

106 It is clear from the existing literature that there is a lack of GA exemplars that could demonstrate
107 the capacity to solve realistic large-scale construction TCT problems efficiently to make them
108 accessible for real life applications. The term efficiency is used here to indicate solutions with
109 reasonable computational requirements that can be met by a personal computer, and processing
110 time (~minutes). The main motivation behind this research is to address this gap, and following
111 are the main research questions:

- 112 • How feasible, in terms of computational demand and accuracy, is the application of meta-
113 heuristics to realistic TCT problems?
- 114 • What features of the TCT problem can be exploited to improve the efficiency of metaheuris-
115 tics?
- 116 • What is the software development effort required to achieve such solutions?

117 Motivated by these research questions, this article provides the following contributions to the
118 construction industry and research community:

- 119 • The presentation of a holistic analysis on the utility of GA to solve real-world TCT problems.
120 This analysis considers the trade-off of solution optimality in terms of time-to-solution, the
121 complexity of implementation and computer hardware/software requirements.
- 122 • The presentation of a number of well-studied, but highly effective, methods for improving
123 GA performance of TCT problems. These include optimization of fitness evaluations using
124 weighted graphs, a population stagnation criterion, and implementing parallel computing.
- 125 • The provision of guidelines for GA parameter selection for TCT problems based on param-
126 eter performance assessment using approximate Bayesian techniques.
- 127 • The demonstration of the efficacy of freely available, standard “off-the-shelf” GA imple-

128 mentations in solving realistic TCT problems typical of real construction projects.

129 **Organization of the Article**

130 The following section of the article describes the research methodology; the benchmark models
131 used, solution method, and implementation approach are discussed in detail. An improved GA
132 design to solve TCT problems is then introduced. This section includes discussions on how to
133 encode the problem, evaluate fitness and using a stopping criteria to maximize the computational
134 efforts. Also described in this section is a novel approach to fitness assessment step of GA
135 development. The results section includes discussions on how to solve the benchmark problems
136 where the model performance is assessed in terms of accuracy, processing time and processing
137 demand. The article concludes with the summary and long term implications of the findings.

138 **RESEARCH METHODOLOGY**

139 **Defining the Problem—Benchmark Studies**

140 The majority of literature reviewed use the fundamental problems defined by [Feng et al. \(1997\)](#)
141 and [Chassiakos and Sakellariopoulos \(2005\)](#). This article is structured around the 18- and 63-
142 variable problems defined by [Feng et al. \(1997\)](#) and [Sonmez and Bettemir \(2012\)](#) in devising larger
143 networks. This was achieved by repeating these network in series to achieve larger networks, and
144 because the solution to smaller networks are known, so are the solutions to the larger networks.
145 This property provides the benchmark for accuracy comparisons. Although the networks created
146 in this fashion may not capture actual complexity of networks of comparable size, this method has
147 been used and accepted in earlier literature consistently ([Aminbakhsh and Sonmez 2016](#)).

148 Algorithm development efforts in this article was structured around the 63-variable networks
149 defined by [Sonmez and Bettemir \(2012\)](#). These two topologically identical networks (63a and 63b)
150 have a minor difference in their respective indirect cost figures. Additionally, results for analyses
151 of larger networks of 630, 1800, 3150 and 6500 variables are also provided. [Sonmez and Bettemir](#)
152 [\(2012\)](#) provide an interesting perspective on the increased computational complexity as the number
153 of time-cost pairs associated with these networks—the computational complexity of these networks

154 are likely to increase exponentially with increased activity numbers. For the 18-variable problem
155 there are 5.9E+9 discrete time-cost alternatives, while the same for the 63-variable problem are
156 1.4E+42. For 180- and 630-variable networks, these values are 5.2E+97 and 2.9E+421 respectively.
157 Note that these values only indicate the mode selection for activities, the topology sorting for larger
158 networks (i.e. Critical path computations) will also be more complicated.

159 **Solution Method—Genetic Algorithms**

160 Genetic Algorithms (GA) have long been the most common and widely used metaheuristics
161 in construction engineering and management research. However, examples of more sophisticated
162 metaheuristics solutions such as Particle Swarm Optimization (Zhang et al. 2005), Shuffled Frog
163 Leaping (Fang and Wang 2012), Ant Colony Optimization (Merkle et al. 2002), and Memetic
164 Algorithms (Senouci and Eldin 2004) also exist (Zhang and Ng 2012). Despite this popularity, no
165 clear benchmarks exist on their design and capacity as potential practical solutions to large-scale
166 TCT problems outside few studies (Aminbakhsh and Sonmez 2016; Menesi et al. 2013; Kandil
167 et al. 2010). Thus, this article provides analysis of GAs performance across different benchmark
168 problems as a starting point to test their performance.

169 **Solution Platform**

170 An important goal with this article was to assess the GA performance without devising purpose
171 specific algorithms in an attempt to minimize the development efforts and ensure extensibility of the
172 proposed solutions to different problems. The application platform chosen for this study was to use
173 a generic GA solution through the Distributed Evolutionary Algorithms in Python (DEAP) frame-
174 work (Fortin et al. 2012). This open source framework is built on the Python programming
175 language—an open source, high-level programming language (<https://www.python.org/>).
176 DEAP was chosen due to the ease of development and integration of parallel computing. The
177 method chosen for parallelization was Scalable COncurrent Operations in Python (SCOOP) within
178 the DEAP framework, due to its seamless integration with DEAP and relatively low "cost of
179 development" in implementation (Hold-Geoffroy et al. 2014).

180 **Preliminary Implementation and Observations**

181 To assess the baseline performance of DEAP as a general implementation method and SCOOP
182 as the parallel computing medium, preliminary optimization instances were run. No modifications
183 were made to the DEAP code to implement the problem to assess its viability for scalability
184 and reduce development efforts. A reasonable assumption was made about the upper limits of
185 the population size (2,000) and the number of generations (2,000) (Kandil and El-Rayes 2006).
186 The default GA parameters were; mutation rate of 0.1 and method of Polynomial Bounded (also
187 available Gaussian, Shuffle Indexes, Flip Bit and Uniform), crossover rate of 0.5 and method
188 of Uniform Partially Matched (also available, One and Two Point, Uniform, Partially Matched,
189 Ordered and Blend), and selection method of NSGA-II (also available, Tournament, Roulette,
190 SPEA-II, Random, Best, Worst, Tournament BCD) (Fortin et al. 2012). A snapshot of the overall
191 performance of the results of these runs is given in Table 2. The reported baseline performance
192 values are for 630-variable problem as this was the most recent benchmark problem, and more
193 detailed analyses for different networks are provided in later parts of this article.

194 Remarkably, the results were extremely accurate, exceeding those reported by Sonmez and
195 Bettemir (2012); however, the computational time, on average, was approximately 18 hours for
196 the five routines run for each one of these test instances. Due to the expect non-linear growth in
197 processing time of the larger network problems, they were not attempted using this method. Having
198 arrived at an accurate DEAP implementation of the benchmark problem by Sonmez and Bettemir
199 (2012), this research was extended to exploit parallel computing using SCOOP. This resulted in
200 more than 4x speedup over the serial code using 8 CPU cores; however, the overall runtime (4 hrs)
201 indicated solving realistic TCT problems would still be infeasible.

202 **IMPROVED GENETIC ALGORITHM DESIGN**

203 The preliminary analysis indicated that the DEAP package can provide accurate solutions;
204 although, the processing times were still significant. Parallel computing with the SCOOP package
205 also reduced the computational time significantly when the computations were distributed over
206 multiple CPUs. Considering these observations, the following sections of this article show further

207 improvements made to the GA design—including introducing a stopping criterion, details of
208 graph-based fitness evaluation method, and selection of optimal GA parameters—to minimize
209 computational time without sacrificing accuracy.

210 **Problem Encoding**

211 Arguably the most important element in effective use of a GA is the encoding of a problem
212 solution as a genetic code. In terms of biological analogy, the problem solution can be considered
213 as the organism or the phenotype and the genetic code is the genotype. The method of reading
214 a genome to produce a problem solution is the embryology. The choice of encoding along with
215 the associated embryology, crossover, mutation and selection operations can have a significant
216 impact on the effectiveness of the method (Affenzeller et al. 2009). A simple encoding approach
217 was determined to be effective in modeling the TCT problem, and the genotype of a individual is
218 defined as a sequence of construction modes.

219 Firstly consider the time-cost optimization problem with n activities $A = \{a_1, a_2, \dots, a_n\}$ where
220 a_i is an integer that represents the number of modes that can be assigned to the i th activity. Each
221 mode assignment, $m_{i,j}$, is a cost/duration pair, $m_{i,j} = (c_{i,j}, d_{i,j})$, where $c_{i,j}$ and $d_{i,j}$ are respectively
222 the cost and duration of the i th activity using the j th mode. The genotype for a potential solution set
223 of mode assignments is given by the list $G = \{g_1, g_2, \dots, g_n\}$ where $1 \leq g_i \leq a_i$ for $i = 1, 2, \dots, n$.
224 Here each gene, g_i , encodes the mode number for activity i .

225 The phenotype is constructed by assigning activity i to the g_i th possible mode. This yields an
226 embryology function $E(G)$ defined by,

$$227 \quad E(G) = \{(1, g_1), (2, g_2), \dots, (n, g_n)\}. \quad (2)$$

228 A fitness function the can be determined in terms of the phenotype and genotype.

229 **Fitness Evaluation**

230 The fitness of a individual phenotype $p = E(G)$, is the total cost of the construction process
231 given the activity mode assignments. By assignment is referred to an activity-mode pair. For

232 example, if activity i will implement mode g_i , then the mode assignment is $p_i = (i, g_i)$. This total
 233 cost, C_T , consists of the sum of the activity costs, C_A , and the indirect costs, C_I , of the total project
 234 duration. That is,

$$235 \quad C_T = C_A + C_I. \quad (3)$$

236 This is simply Equation (1) without the incentive and disincentive terms—it should be noted that
 237 these terms can be simply added to the formula if the problem characteristics requires so.

Given individual, p , C_A and C_I can be calculated as,

$$C_A = \sum_{i=1}^n c_{p_i}, \quad (4)$$

$$C_I = r \sum_{j \in L} d_{p_j}. \quad (5)$$

238 where r is the indirect cost rate (\$/time) and L is the set of activities which contribute to the critical
 239 path of the project.

240 By substitution of Equations (4) and (5) into Equation (3) a fitness function can be defined,

$$241 \quad F(p) = \sum_{i=1}^n c_{p_i} + r \sum_{j \in L} d_{p_j}. \quad (6)$$

242 This can, in turn, be expressed in terms of the genotype using the embryology (Equation (2)),

$$243 \quad F^*(G) = \sum_{i=1}^n c_{i,g_i} + r \sum_{j \in L} d_{j,g_j}. \quad (7)$$

244 Computational complications arise from computing the latter part of the formula caused by the
 245 necessity to compute the longest path (i.e. critical path) of the network analyzed. A significant
 246 contribution of this article to general construction engineering and management literature is the
 247 computation of the critical path using a robust and efficient method from graph theory. This method
 248 removes the need for repeated topology sorting with different time-cost mode allocation and improve
 249 overall computation time substantially. Used in various other disciplines such as Operations

250 Management (e.g. minimum spanning tree), graph theory is the mathematics of objects with
 251 interconnected relationships (Floudas and Pardalos 2008). Although graph theory has been applied
 252 to general scheduling problems (Dharwadker and Pirzada 2011), no such study in construction
 253 engineering and management literature exists to the best of the authors knowledge. The main
 254 advantage of using a graph-based solution to the classical TCT problem is the requirement for
 255 topological sorting of the network needs only be completed once, rather than more traditional
 256 approach of iteratively constructing the critical path for each one of the GA solutions. Because
 257 the time-cost pairs are assigned to activities in no particular order in search of the lowest project
 258 costs, all corresponding topologies needs to be sorted (i.e. Critical path) at each step of genetic
 259 operations. It is not clear whether this iterative approach was replicated in earlier literature, but this
 260 method was used in the earlier optimization analyses conducted for this article. This also explains
 261 the significant computational time associated with these optimization runs.

262 **Stopping Criteria**

263 A challenge in using GA is to determine when to terminate (Affenzeller et al. 2009; Kim
 264 2013) the genetic operations. This is a trade-off between the quality of the solution and the overall
 265 execution time. Everything kept equal, more generations will always lead to better solutions unless
 266 the correct answer is found during computations; however, the rate at which this improvement occurs
 267 is greatly dependent on the genetic diversity in the population of solutions. When the diversity of
 268 the solutions gets low, the populations stagnate and the solutions are essentially stabilized. At this
 269 point there is little to be gained by continued genetic operations, and it is sensible to terminate the
 270 evolution to optimize computation time required. To achieve this, a measure of diversity, which
 271 compares the relative distance between fitness of the best individual to the average fitness of the
 272 whole population was devised.

$$273 \quad S(P) = 1 - |P| \frac{\min_{G \in P} F^*(G)}{\sum_{G \in P} F^*(G)} \quad (8)$$

274 where P is the gene pool, that is the set of all genomes in the current population. This stagnation

275 criterion is used to determine the trade-off between computational effort and solution accuracy. In
276 this article, the stagnation threshold of $T = 5E - 005$, and the stopping criterion of $S(P) < T$ were
277 used.

278 **Critical Path Computations**

279 The total cost function (Equation (3)) and derived fitness function (Equation (7)) consist of
280 two distinct components; the activity costs and the indirect cost. Of these, the indirect cost is
281 more computationally expensive due to the requirement that the critical path (i.e. duration) of
282 the project must be calculated to determine the total project duration. The reviewed construction
283 engineering and management literature have not explicitly stated the method used in computing
284 the project duration; but, it appears this is computed as the longest path within a network (Sonmez
285 and Bettemir 2012). This can simply be calculated by iteratively computing the completion time
286 of activities using the precedence logic. In this article, the preliminary computations were run
287 using this method. Once an individual activity time-cost pair is selected from available modes
288 for an activity, the time is registered as the accepted activity duration and fed into the critical
289 path computations. This is identical to the forward pass computations of traditional CPM method.
290 Although effective in providing a simple solution to project duration computations, this approach
291 proves to be inefficient due to excessive time required to compute the results, as in each iteration
292 the network topology needs to be recalculated because of the activity mode selection.

293 *The Longest Path Problem*

294 Critical path problem is a specific case of a *longest path problem*. This widely studied fun-
295 damental problem of finding the longest path within a graph is essentially the very description
296 to finding the longest path within a given schedule (Floudas and Pardalos 2008). Studying the
297 effective solution algorithms devised from the longest path problem, a computationally efficient
298 solution for the TCT problems under consideration was developed.

299 *Graph Method*

300 A graph is defined as a set of vertices, V , and a set of edges, E . For an undirected graph, vertices
301 $u, v \in V$ are said to be *connected* if there is an edge $e \in E$ such that $e = (u, v)$. The *neighborhood* of
302 vertex v by $n(v)$, that is, $n(v)$ is the set of all vertices in E that are connected to v . A weighted graph
303 also includes a function $\omega(e)$ which assign edges to a measure of distance between its endpoints.
304 Importantly, a mode's duration through the weighting function $\omega(e)$ was explicitly encoded, which
305 enables efficient pre-processing via a topological sort.

306 Given a path is defined as a connected sequence of vertices $P_{v_0, v_n} = (v_0, v_1, \dots, v_n)$, that is for all
307 $i \in [0, n - 1]$ there is an edge $(v_i, v_{i+1}) \in E$. The length of a path, denoted by $D(P_{v_0, v_n})$, is the
308 sum of the edge weights connecting the vertices in the path, that is $D(P_{v_0, v_n}) = \sum_{i=0}^{n-1} \omega((v_i, v_{i+1}))$.

309 The longest path problem can be stated as follows: Given a weighted graph, $G = (V, E)$, with
310 weighting function $\omega(e)$ and two vertices $s, t \in V$, the task is to find a path $L_{s,t}$ that satisfies,

311
$$L_{s,t} = \operatorname{argmax}_{P_{s,t} \in \Phi_{s,t}} D(P_{s,t}), \tag{9}$$

312 where $\Phi_{s,t}$ is the set of all possible paths between s and t . It should be noted that for a general
313 graph G the longest path problem has been shown to be NP-hard—similar to the TCT problem as
314 noted by [Sonmez and Bettemir \(2012\)](#).

315 *Solution for Directed Acyclic Graphs*

316 There are two special types of graphs: directed and acyclic graphs. A directed graph defines
317 connectivity as unidirectional, that is if $(u, v) \in E$ then u is connected to v but v is not connected
318 to u . A graph is acyclic if there do not exist and paths with $v_0 = v_n$, such a path is called a cycle.
319 Although the longest path problem is NP-hard for a general weighted graph, an efficient algorithm
320 exists for graphs which are directed and acyclic with a non-negative weight function $\omega(e) \geq 0$ for
321 all $e \in E$ ([Kahn 1962](#); [Knuth 1968](#)). The algorithm consists of two steps: (1) sort the vertices
322 topologically, and (2) visit vertices in order and compute longest path as the longest path from each
323 incoming edge.

324 A vertex list of a directed acyclic graph (DAG) is topologically sorted if for every directed edge
325 $e = (v_i, v_j)$ then $i < j$. That is, all vertices from all possible paths leading to the j -th vertex are
326 visited before the j -th vertex itself. A topological sort can be computed using the algorithm shown
327 in Figure 1. It is worth noting that a topological sort is only valid for directed acyclic graphs. Once
328 the topological sort has been applied, the longest path between all nodes can be computed using
329 the algorithm shown in Figure 2.

330 This longest path algorithm has a computational complexity of $O(|V| + |E|)$. Since the activity
331 dependency graph for a TCT problem is *directed, acyclic and activity durations are always non-*
332 *negative*, the algorithm shown in Figure 2 is directly applicable.

333 Thus, the fitness function $F^*(g)$ (Equation (7)) can be computed efficiently in $O(|V| + |E|)$
334 operations (Cormen et al. 2001). Figure 3 demonstrates the relative proportion of CPU time
335 spent in fitness evaluations using the graph-based scheme versus the original, iterative approach.
336 This figure shows computational demands for different parts of the overall computations and the
337 advantage of DAG method over the traditional approach to the 63a problem.

338 **DAG Implementation Using DEAP and SCOOP**

339 A parallel genetic algorithm for solving the TCT using Python was implemented, in which
340 the computationally demanding critical path durations were calculated using the graph theory.
341 The DEAP library was utilized for the genome definition population creation and genetic opera-
342 tions (Fortin et al. 2012). The fitness evaluation step, which is the most computational task, has
343 been parallelized using the SCOOP library (Hold-Geoffroy et al. 2014).

344 At the beginning of each iteration, individuals were randomly paired together based on fitness
345 based on one of several selection methods. Crossover is then performed with each of these pairs
346 to produce exactly two offspring, upon which mutation is applied at random. Finally, all offspring
347 and parents are pooled into one population, from which the best 50% are selected for the next
348 generation. This cycle repeats until a fixed generation limit is reached or if the population stagnates
349 (As computed by Equation 8). An overview of this approach is given in Figure 4.

350 The most efficient method of computing the longest path in a DAG consists of a topological sort

351 followed by an incremental build of the longest path by choosing the longest incident path at each
352 node. Because individual activity dependencies do not change with different activity duration-cost
353 pair selection, the topological sort needs to be applied once. This dramatically increases the overall
354 performance of the critical path computations. Larger problems were solved efficiently by removing
355 repeated topological sorting. Since each fitness evaluation is completely independent, these can be
356 executed in parallel using the Python SCOOP library (Hold-Geoffroy et al. 2014).

357 **Parameter Selection**

358 Performance (computation time and accuracy) of GAs are affected by the parameter selection,
359 and to the best of authors' knowledge, no comprehensive parameter sweep of GA parameters has
360 been conducted on TCT problem solution criteria—examples of such studies exist for different
361 scheduling problems (Alcaraz and Maroto 2001). To address this research gap, *approximate*
362 *Bayesian computation* (ABC) techniques were used to analyse the algorithm parameter distribution
363 for a given desired accuracy threshold. The vector of algorithm parameters, θ , was treated as a
364 random variable with probability distribution $p(\theta)$, then the parameter distribution was computed,
365 in which the solution under a given parameter set, S_θ , is the exact minimum solution, S_M using
366 Bayes' Theorem,

$$367 \quad p(\theta \text{ given } S_\theta = S_M) = \frac{p(S_\theta = S_M \text{ given } \theta)p(\theta)}{p(S_\theta = S_M)}. \quad (10)$$

368 While Equation (10) is not tractable, it can be estimated for a tolerable level of accuracy, ϵ ,
369 through GA simulations with randomized parameters. This method, given in Figure 5, that results
370 is know as ABC rejection sampling in the computational statistics literature (Sunnaaker et al. 2013).

371 This ABC rejection scheme was applied using the 63a and 63b TCT problems using a target
372 accuracy of $\epsilon = 0.7\%$ with uniform initial parameter distributions. For the crossover and mutation
373 rates this was continuous uniform distributions in the interval $[0, 1]$, and the crossover, mutation
374 and selection functions were treated as uniform discrete random variables (i.e., each was selected
375 with equal probability from the available options within DEAP documentation). These values were

376 used for the limits for the initial parameter distribution for ABC analysis. After computing 300
377 accepted ABC samples, it was determined that the most optimum parameters in balancing accuracy
378 and processing time using the mean of the ABC samples. For the selection, crossover and mutation
379 functions, this mean was rounded and found the *Tournament* selection function, the *One Point*
380 cross-over function, and the *Shuffle Indexes* mutation function to be optimal. The optimal mutation
381 and crossover rates were found to be 0.107 ± 0.079 and 0.474 ± 0.082 respectively. Note that
382 the ABC analyses in this article exclude number of generations run and the population size as they
383 were discussed in earlier parts of this article. A particularly interesting observation is the lack of
384 performance from widely popular functions such as NSGA-II and SPEA-II. This can be explained
385 in part that these algorithms are suitable for multi-objective optimization, and the TCT problems
386 presented here are not and can be solved with simpler and faster converging algorithms.

387 **RESULTS AND DISCUSSIONS**

388 **Performance**

389 *Solution Accuracy*

390 The largest network with clear accuracy numbers has been provided by Sonmez and Bettemir
391 (2012) as 2.41% and 2.47% for 630a and 630b problems using a hybrid GA the authors have
392 developed. The algorithms used in this research produced significantly better results for the same
393 networks during parameter selection analyses described in earlier sections of this article. The
394 average accuracies were 0.79% (min=0.43%, max=1.57%) & 1.12% (min=0.64%, max=1.71%)
395 for 630a and 630b problems respectively. The results were encouraging and clearly indicate
396 the efficacy of the proposed optimization routine in achieving extremely accurate solutions to a
397 large scale TCT problem. However, for any solution of this problem to be practically viable,
398 accuracy needs to be accompanied by convergence speed, as well as development effort—a factor
399 mostly ignored in earlier literature. Because of these encouraging results and significantly faster
400 convergence rates, larger networks of sizes 1800, 3150 and 6300 variables were solved to test the
401 convergence limits of the algorithms.

402 No stagnation criteria was used in these experiments as the goal was to assess the limits of
403 convergence, rather than practical solutions. The analyses were run five times for each one of these
404 instances and averages values for both accuracy and runtime are reported in Table 3. Number of
405 generations was set to 5,000 with a population size of 2,750 to test the accuracy limits of the proposed
406 solution. These test instances were run with 8 CPUs. The results of these larger networks are also
407 encouraging with the exception of an 1800-variable network. There are no clear explanations to why
408 this variability of the accuracies observed while processing times were comparable to the network
409 sizes. Another interesting observation is the growth in processing time appears to be near linear
410 with increased network size. This is a clear indicator of the efficiency of the proposed method
411 in addressing the computational complexity of these problem under assessment. The observed
412 variable accuracy might have been caused by the inherent variability of the results consistency of
413 metaheuristic methods for different problems or the complexities of solution surfaces of the original
414 18-variable problem might be more complicated than those of the 63-variable problem; although,
415 the latter is the much larger network.

416 *Development Effort*

417 It is common in construction engineering and management research to use existing, open source
418 optimization routines (Kandil and El-Rayes 2006), purpose specific software platforms (Menesi
419 et al. 2013), or develop customized optimization routines (Sonmez and Bettemir 2012) using
420 different software packages and programming languages. In this article, DEAP, and open source
421 development project aimed at increasing the viability of Evolutionary Algorithms written in Python,
422 was used. In developing solutions for the TCT and parallelism, the main goal was to minimize
423 the development efforts and rely as much on off-the shelf solutions. Similar to GA development,
424 SCOOP package was used for parallel computing, which was in built to DEAP package with
425 minimal alterations.

426 *Runtime Performance and Processing Demand*

427 Kandil and El-Rayes (2006) cited a TCT problem of 720-variable network that took 136.5 h
428 to solve, which was reduced to 19.72 h using parallel computing and 6.70 h when using coarse-

429 grained parallel modules. In a follow up article, the authors reported (Kandil et al. 2010) a 90
430 h runtime without parallel computing that can possibly be attributed to faster processors. The
431 authors also provide a processing time versus number of processors to evaluate their relationship.
432 Following this approach, a comparison of reduced processing time with added computational
433 resources through parallel computing is provided. Table 4 summarizes the runtime performance
434 with added computational demand across multiple levels of accuracy. It can be seen that problems
435 of 630 variables are solved under 10 minutes consistently using no more than 8 CPUs. This is
436 significant as these results indicate a more than 100× speed increase to the non-graph, non-parallel
437 GA approach used earlier in this article (see Table 2).

438 In parallel computing, the limitations of parallelization must also be considered. It is important
439 to note that the maximum speed up of a parallel algorithm is constrained by the ratio of strictly serial
440 portions of the code. This is generally referred to as Amdahl's Law (Amdahl 1967). Essentially,
441 this is an example of the law of diminishing returns. As the parallel portion of the code is increased,
442 the strictly serial portion becomes a bigger proportion of the total runtime. As the the number of
443 cores goes to infinity, then the limit of the speed up is the original serial code runtime divided
444 by the strictly serial portion of the parallel code—theoretically this is the maximum possible
445 parallelization. During the analyses, CPUs of 1-8 were considered. In majority of these runs, no
446 noticeable improvements in either time or accuracy were noted when more than 6 CPU's where
447 used. The algorithms for this research effort were developed using a state of art HPC facility, which
448 brings the questions about how viable this method for implementation on a personal computer. To
449 test the viability of solving these problems in a personal computer, some of these test instances were
450 also run on a personal computer. This PC had a 6 core Intel Xeon CPU (W3670 clocked at 3.2 GHz)
451 with 16 GB of RAM and the operating system was the Red Hat Enterprise Linux version 6.4. This
452 is a relatively old (~2010) system that performed on par with the HPC runs for larger problems,
453 indicating that common PCs can be used in applications based on the methodology proposed here.

454 **Strengths, Weaknesses of the Proposed Solution**

455 The DAG model implemented in DEAP and SCOOP can be a viable solution to large scale
456 discrete TCT problems that are common in the construction industry. The implemented model was
457 developed with minimum coding and software development using open source tools, and processing
458 demands can be met by a personal computer while keeping the computational time manageable.
459 The solution developed in this article compares favorably in terms of accuracy and processing
460 time when compared to the earlier studies. Development effort comparisons are more challenging
461 as specifics of earlier studies are not detailed to provide such a comparison. It can be deduced
462 from the earlier articles that researchers have developed problem specific GA solutions (Sonmez
463 and Bettemir 2012), and built on existing algorithms (Kandil et al. 2010); however, there are no
464 discussions on the development efforts needed. The parameter sweep analyses conducted indicates
465 that the model performance will be heavily dependent on problem formulation and GA parameter
466 selection. This is, perhaps, also the biggest weakness of the algorithms developed in this article as
467 their performance is not consistent across different problems and users will have to make decisions
468 on what parameters should be selected. The proposed solution here is a steep improvement from
469 traditional GA solutions to the TCT problem; however, there are other solutions that can solve
470 the TCT problem (Aminbakhsh and Sonmez 2016). The solution set presented in this article is
471 not mutually exclusive to other efforts as the DAG method can be implemented in longest path
472 computation step of different algorithms.

473 **CONCLUSION**

474 This article presents an innovative approach to solving large scale construction time-cost trade-
475 off problems using genetic algorithms (GAs) that can solve real life scale TCT problems with
476 exceptional accuracy, while not creating excessive computational demand or processing time. This
477 is achieved by using a well-studied, yet innovative method of graph theory in solving the critical
478 path problem. The computational bottleneck in solving large scale TCT problems is the fitness
479 assessment of different solutions because of the iterative and repeated computation of critical paths
480 (i.e. the longest path in a network). The graph method was used to carry out topology sorting

481 one time, which increases computational efficiency significantly. Moreover, this research effort
482 implemented parallel-computing, provide an approximate Bayesian assessment of optimum GA
483 parameters, and introduce an effective stopping criteria; all of which further improves computational
484 efficiency. The methods provided here represent the groundwork for analyzing even larger networks,
485 and can be extended to different construction engineering and management problems such as
486 resource constrained scheduling and resource leveling. Because the methodological improvements
487 proposed here are non domain-specific, they can be used to efficiently solve different computational
488 problems.

489 **ACKNOWLEDGMENTS**

490 Computational resources and services used in this work were provided by the High Performance
491 Computing (HPC) and Research Support Group of Queensland University of Technology, Brisbane,
492 Australia. The authors also like to thank Dr Justin Lee for his technical support in Genetic Algorithm
493 design.

REFERENCES

- Affenzeller, M., Winkler, S., Wagner, S., and Beham, A. (2009). *Genetic Algorithms and Genetic Programming*. CRC Press.
- Alcaraz, J. and Maroto, C. (2001). “A robust genetic algorithm for resource allocation in project scheduling.” *Annals of Operations Research*, 102(1-4), 83–109.
- Amdahl, G. M. (1967). “Validity of the single processor approach to achieving large scale computing capabilities.” *Proceedings of the April 18-20, 1967, spring joint computer conference*, ACM, 483–485.
- Aminbakhsh, S. and Sonmez, R. (2016). “Discrete particle swarm optimization method for the large-scale discrete time–cost trade-off problem.” *Expert Systems with Applications*, 51, 177 – 185.
- Boussaïd, I., Lepagnot, J., and Siarry, P. (2013). “A survey on optimization metaheuristics.” *Information Sciences*, 237, 82–117.
- Chassiakos, A. P. and Sakellariopoulos, S. P. (2005). “Time-cost optimization of construction projects with generalized activity constraints.” *Journal of Construction Engineering and Management*, 131(10), 1115–1124.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. Cambridge, MA: MIT Press.
- Debels, D. and Vanhoucke, M. (2007). “A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem.” *Operations Research*, 55(3), 457–469.
- Dharwadker, A. and Pirzada, S. (2011). *Applications of Graph Theory*. CreateSpace Independent Publishing Platform (1 October).
- Fang, C. and Wang, L. (2012). “An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem.” *Comput. Oper. Res.*, 39(5), 890–901.
- Feng, C.-W., Liu, L., and Burns, S. A. (1997). “Using genetic algorithms to solve construction time-cost trade-off problems.” *Journal of Computing in Civil Engineering*, 11(3), 184–189.
- Floudas, C. A. and Pardalos, P. M. (2008). *Encyclopedia of optimization*, Vol. 1. Springer Science

521 & Business Media.

522 Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). “DEAP:
523 Evolutionary algorithms made easy.” *Journal of Machine Learning Research*, 13, 2171–2175.

524 Gaglioppa, F., Miller, L. A., and Benjaafar, S. (2008). “Multitask and multistage production
525 planning and scheduling for process industries.” *Operations Research*, 56(4), 1010–1025.

526 Ghoddousi, P., Eshtehardian, E., Jooybanpour, S., and Javanmardi, A. (2013). “Multi-mode
527 resource-constrained discrete time–cost-resource optimization in project scheduling using non-
528 dominated sorting genetic algorithm.” *Automation in Construction*, 30, 216 – 227.

529 Hartmann, S. and Briskorn, D. (2010). “A survey of variants and extensions of the resource-
530 constrained project scheduling problem.” *European Journal of Operational Research*, 207(1), 1
531 – 14.

532 He, Z., He, H., Liu, R., and Wang, N. (2017). “Variable neighbourhood search and tabu search for
533 a discrete time/cost trade-off problem to minimize the maximal cash flow gap.” *Computers &
534 Operations Research*, 78, 564 – 577.

535 Hegazy, T. (1999). “Optimization of construction time-cost trade-off analysis using genetic algo-
536 rithms.” *Canadian Journal of Civil Engineering*, 26(6), 685–697.

537 Hold-Geoffroy, Y., Gagnon, O., and Parizeau, M. (2014). “Once you scoop, no need to fork.”
538 *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery
539 Environment*, ACM, 60.

540 Kahn, A. B. (1962). “Topological sorting of large networks.” *Commun. ACM*, 5(11), 558–562.

541 Kandil, A. and El-Rayes, K. (2006). “Parallel genetic algorithms for optimizing resource utilization
542 in large-scale construction projects.” *Journal of Construction Engineering and Management*,
543 132(5), 491.

544 Kandil, A., El-Rayes, K., and El-Anwar, O. (2010). “Optimization research: Enhancing the ro-
545 bustness of large-scale multiobjective optimization in construction.” *Journal of Construction
546 Engineering and Management*, 136(1), 17–25.

547 Kim, J.-L. (2013). “Genetic algorithm stopping criteria for optimization of construction resource

scheduling problems.” *Construction Management and Economics*, 31(1), 3–19.

Kim, J.-L. and Ellis Jr, R. D. (2008). “Permutation-based elitist genetic algorithm for optimization of large-sized resource-constrained project scheduling.” *Journal of Construction Engineering and Management*, 134(11), 904–913.

Knuth, D. E. (1968). *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley.

Long, L. D. and Ohsato, A. (2009). “A genetic algorithm-based method for scheduling repetitive construction projects.” *Automation in Construction*, 18(4), 499–511.

Menesi, W., Golzarpoor, B., and Hegazy, T. (2013). “Fast and near-optimum schedule optimization for large-scale projects.” *Journal of Construction Engineering and Management*, 139(9), 1117–1124.

Merkle, D., Middendorf, M., and Schmeck, H. (2002). “Ant colony optimization for resource-constrained project scheduling.” *IEEE Trans. Evol. Comput.*, 6(4), 333–346.

Michalewicz, Z. (1999). *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, Berlin; New York.

Monghasemi, S., Nikoo, M. R., Fasaee, M. A. K., and Adamowski, J. (2015). “A novel multi criteria decision making model for optimizing time–cost–quality trade-off problems in construction projects.” *Expert Systems with Applications*, 42(6), 3089 – 3104.

Senouci, A. B. and Eldin, N. N. (2004). “Use of genetic algorithms in resource scheduling of construction projects.” *Journal of Construction Engineering and Management*, 130(6), 869–877.

Sonmez, R. and Bettemir, Ö. H. (2012). “A hybrid genetic algorithm for the discrete time–cost trade-off problem.” *Expert Systems with Applications*, 39(13), 11428–11434.

Sunnaker, M., Busetto, A. G., Numminen, E., Corander, J., Foll, M., and Dessimoz, C. (2013). “Approximate bayesian computation.” *PLOS Computational Biology*, 9(1), 1–10.

Tran, D.-H., Cheng, M.-Y., and Prayogo, D. (2016). “A novel multiple objective symbiotic organisms search (mosos) for time–cost–labor utilization tradeoff problem.” *Knowledge-Based Systems*, 94,

- 575 132 – 145.
- 576 Vanhoucke, M. and Debels, D. (2007). “The discrete time/cost trade-off problem: extensions and
577 heuristic procedures.” *Journal of Scheduling*, 10(4), 311–326.
- 578 Zhang, H. (2011). “Ant colony optimization for multimode resource-constrained project schedul-
579 ing.” *Journal of Management in Engineering*, 28(2), 150–159.
- 580 Zhang, H., Li, X., Li, H., and Huang, F. (2005). “Particle swarm optimization-based schemes for
581 resource-constrained project scheduling.” *Autom. Constr.*, 14(3), 393–404.
- 582 Zhang, Y. and Ng, S. T. (2012). “An ant colony system based decision support system for construc-
583 tion time-cost optimization.” *Journal of Civil Engineering and Management*, 18(4), 580–589.

584
585
586
587
588
589

List of Figures

1 Topological Sort 30

2 Longest Path Algorithm for a Topological Sorted, Non-Negatively Weighted DAG . 31

3 Computational Demand Profile for GA Implementation 32

4 Flow Chart of GA Implementation 33

5 ABC rejection sampling for optimal algorithm parameter selection 34

TABLE 1. Sample of Earlier Research Findings on TCT Problem Solution Benchmarks

| Article | Network | Cost-Time Pairs | Method | Runtime(h) | Accuracy(%) | Parallelism |
|--------------------------------------|---------|-----------------|--------------|------------|-------------|-------------|
| Feng et al. 1997 | 18 | 5 | GA | N/A | N/A | No |
| Hegazy 1999 | 18 | 5 | GA | 0.11 | N/A | No |
| Chassiakos & Sakellaropolous 2005 | 29 | 3 | LP/IP | <0.01 | 0.30 | No |
| Kandil and El Reyes. 2006 | 720 | N/A | GA | 6.70 | N/A | Yes |
| Long & Ohsato 2009 | 18 | 5 | GA | N/A | N/A | No |
| Menesi et al. 2012 | 2000 | 5 | Const. Prog. | 2.00 | 6.39 | No |
| Sonmez & Bettemir 2012 | 630 | 5 | Hybrid GA | 1.22 | 2.41 | No |
| Ghoddousi et al. 2013 | 37 | 2 | GA | N/A | N/A | No |
| Monghasemi et al. 2015 | 18 | 5 | GA | N/A | N/A | No |
| Tran et al. 2016 | 15 | 3 | Hybrid GA | N/A | N/A | No |

TABLE 2. Preliminary GA Benchmarks with DEAP and SCOOP

| Size | DEAP Runtime(h) | DEAP Accuracy(%) | DEAP+SCOOP Runtime(h) | DEAP+SCOOP Accuracy(%) |
|------|-----------------|------------------|-----------------------|------------------------|
| 630a | 17.45 | 0.43 | 4.05 | 0.80 |
| 630b | 18.98 | 0.83 | 4.61 | 1.20 |

TABLE 3. Processing Time and Solution Accuracy for Very Large Networks

| Network | Runtime(h) | Accuracy(%) |
|---------|------------|-------------|
| 1800a | 5.82 | 7.05 |
| 1800b | 5.84 | 14.72 |
| 3150a | 9.15 | 6.50 |
| 3150b | 9.41 | 4.73 |
| 6300a | 16.42 | 7.66 |
| 6300b | 16.76 | 6.96 |

TABLE 4. Runtime & Accuracy vs Number of CPU's

| Stopping Criteria | Network | CPUs | Runtime(m) | Accuracy(%) |
|---------------------|---------|------|-------------|-------------|
| 2,000 Generations | 63a | 2 | 29.04±0.31 | 0.28±0.26 |
| 2,000 Generations | 63a | 4 | 20.74±0.65 | 0.34±0.14 |
| 2,000 Generations | 63a | 6 | 18.93±0.38 | 0.33±0.38 |
| 2,000 Generations | 63a | 8 | 18.32±0.17 | 0.48±0.53 |
| 2,000 Generations | 63b | 2 | 29.01±0.73 | 0.75±0.31 |
| 2,000 Generations | 63b | 4 | 21.88±2.78 | 0.96±0.18 |
| 2,000 Generations | 63b | 6 | 19.06±0.32 | 0.89±0.44 |
| 2,000 Generations | 63b | 8 | 18.39±0.13 | 1.20±0.37 |
| 2,000 Generations | 630a | 2 | 125.72±5.38 | 1.47±0.25 |
| 2,000 Generations | 630a | 4 | 99.38±11.80 | 1.62±0.25 |
| 2,000 Generations | 630a | 6 | 77.14±2.24 | 1.78±0.31 |
| 2,000 Generations | 630a | 8 | 73.92±0.07 | 1.93±0.61 |
| 2,000 Generations | 630b | 2 | 123.25±2.08 | 1.84±0.47 |
| 2,000 Generations | 630b | 4 | 94.72±9.78 | 2.20±0.47 |
| 2,000 Generations | 630b | 6 | 76.76±1.86 | 2.30±0.25 |
| 2,000 Generations | 630b | 8 | 71.99±1.79 | 2.47±0.17 |
| Stagnation | 63a | 2 | 2.15±0.17 | 0.27±0.18 |
| Stagnation | 63a | 4 | 2.06±0.28 | 0.39±0.33 |
| Stagnation | 63a | 6 | 1.91±0.18 | 0.37±0.25 |
| Stagnation | 63a | 8 | 1.85 ±0.13 | 0.26±0.18 |
| Stagnation | 63b | 2 | 2.00±0.37 | 0.93±0.64 |
| Stagnation | 63b | 4 | 1.38±0.42 | 1.84±1.92 |
| Stagnation | 63b | 6 | 1.73±0.26 | 0.64±0.32 |
| Stagnation | 63b | 8 | 1.48±0.23 | 1.24±0.59 |
| Stagnation | 630a | 2 | 10.08±6.15 | 3.15±3.44 |
| Stagnation | 630a | 4 | 9.38±4.13 | 2.25±1.49 |
| Stagnation | 630a | 6 | 5.11±3.34 | 4.37±3.11 |
| Stagnation | 630a | 8 | 6.07±2.27 | 2.76±1.64 |
| Stagnation | 630b | 2 | 8.93±2.93 | 2.49±0.83 |
| Stagnation | 630b | 4 | 6.63±1.38 | 2.63±0.48 |
| Stagnation | 630b | 6 | 4.73±2.80 | 3.77±2.60 |
| Stagnation | 630b | 8 | 6.61±0.77 | 2.29±0.41 |
| Literature Accuracy | 63a | 2 | 1.65±0.10 | 1.39±0.25 |
| Literature Accuracy | 63a | 4 | 1.55±0.06 | 1.33±0.12 |
| Literature Accuracy | 63a | 6 | 1.54±0.13 | 1.40±0.17 |
| Literature Accuracy | 63a | 8 | 1.39±0.08 | 1.50±0.24 |
| Literature Accuracy | 63b | 2 | 1.68±0.28 | 1.42±0.15 |
| Literature Accuracy | 63b | 4 | 1.46±0.12 | 1.65±0.28 |
| Literature Accuracy | 63b | 6 | 1.50±0.14 | 1.50±0.24 |
| Literature Accuracy | 63b | 8 | 1.38±0.19 | 1.50±0.25 |
| Literature Accuracy | 630a | 2 | 10.52±1.10 | 2.18±0.15 |
| Literature Accuracy | 630a | 4 | 7.63±0.64 | 2.18±0.10 |
| Literature Accuracy | 630a | 6 | 7.10±0.63 | 2.23±0.06 |
| Literature Accuracy | 630a | 8 | 7.37±1.26 | 2.33±0.30 |
| Literature Accuracy | 630b | 2 | 10.36±0.98 | 2.30±0.13 |
| Literature Accuracy | 630b | 4 | 7.78±1.13 | 2.32±0.25 |
| Literature Accuracy | 630b | 6 | 7.09±1.78 | 2.47±0.33 |
| Literature Accuracy | 630b | 8 | 6.13±0.58 | 2.43±0.24 |

Note that each of these test instances are run five times, and what is reported in runtime and accuracy columns is the average of these runs and half of the range (max-min) for each. This is done to indicate the approximate distances from the mean for each one of these runs.

```

input : Directed Acyclic Graph  $D = (V, E)$ 
output: Directed Acyclic Graph  $D' = (V', E)$ , such that  $V'$  is topologically sorted
 $V' = \emptyset$ ;
 $Q = \{v : v \in V, (\forall u \in V, (u, v) \notin E)\}$ ;
while  $|Q| \neq 0$  do
   $v \in Q$ ;
   $Q \leftarrow Q - v$ ;
  if  $v \notin V'$  then
     $V' \leftarrow V' \cup \{v\}$ ;
  end
  for  $u$  such that  $(v, u) \in E$  do
    if  $(x, u) \in E$  implies  $x \in V'$  then
       $V' \leftarrow V' \cup u$ ;
    end
  end
end

```

Fig. 1. Topological Sort

input : Directed Acyclic Graph $D = (V, E)$ with weight function $\omega : E \rightarrow \mathbb{N}$
output: Array of predecessors V_p in the longest path for each vertex $v \in V$
for $v \in V$ **do**
 | $V_p[v] \leftarrow \operatorname{argmax}_{u \in \mathcal{P}(v)} d[u] + \omega(u, v);$
 | $d[v] \leftarrow \max_{u \in \mathcal{P}(v)} d[u] + \omega(u, v);$
end

Fig. 2. Longest Path Algorithm for a Topological Sorted, Non-Negatively Weighted DAG

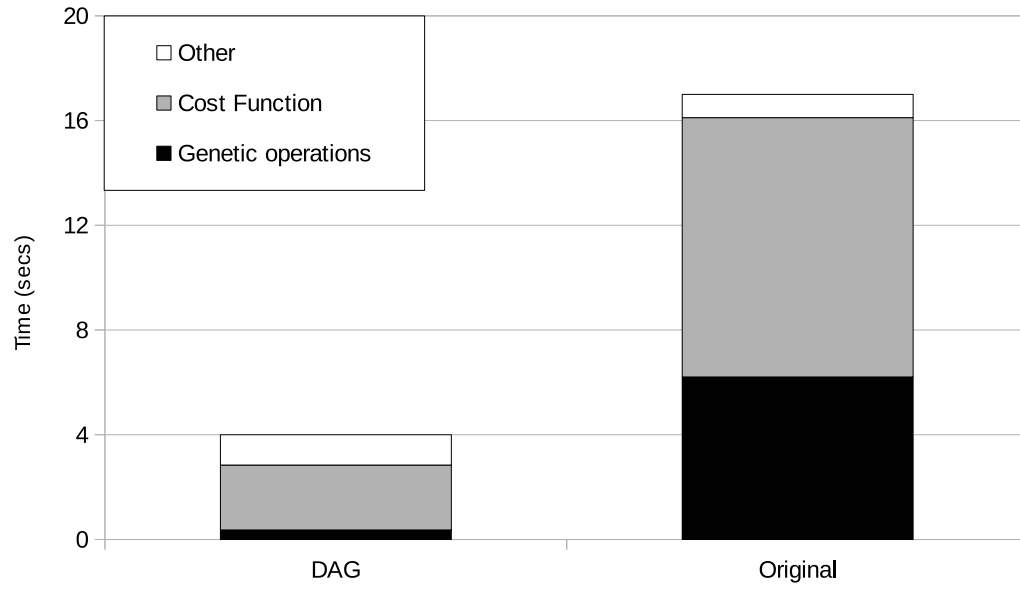


Fig. 3. Computational Demand Profile for GA Implementation

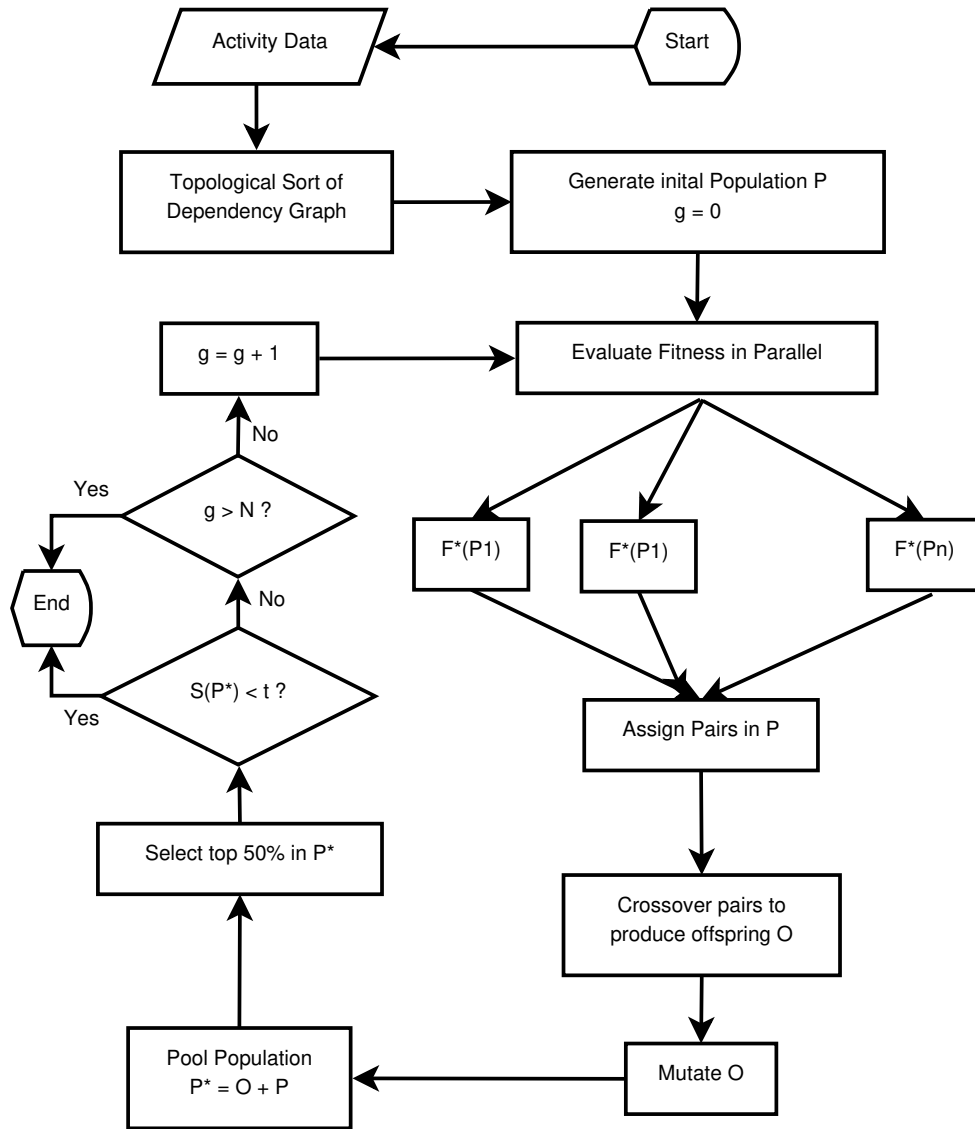


Fig. 4. Flow Chart of GA Implementation

input : Initial parameter distribution $p(\theta)$ and accuracy tolerance ϵ
output: Generates samples, $\theta^1, \theta^2, \dots, \theta^N$, of the parameter distribution with accuracy ϵ
for $i = 1, 2, \dots, N$ **do**
 repeat
 Generate sample of θ^* using $p(\theta)$;
 Compute TCT solution, C_T , using GA algorithm with parameters θ^* ;
 until $C_T \leq \epsilon$;
 $\theta^i \leftarrow \theta^*$;
end

Fig. 5. ABC rejection sampling for optimal algorithm parameter selection