Journal of Engineering Design
Vol. 00, No. 00, Month 2009, 1–27

RESEARCH ARTICLE

Optimising the Time-Based Design Structure Matrix using a Divide and Hybridise Algorithm

Ian Cook and Graham Coates

School of Engineering and Computing Sciences, Durham University, Durham, United Kingdom

(*Received 00 Month 200x; final version received 00 Month 200x*)

Product design and development processes consist of inter-related activities required to be undertaken in an appropriate sequence to reduce the need for iteration and increase opportunities for concurrency. The Design Structure Matrix (DSM) is one of several modelling tools used to represent activities and the dependencies between them. Many algorithms have been developed and applied to the DSM with the aim of determining a near-optimal sequence of activities in terms of a range of objectives. In this paper, an enhanced Genetic Algorithm (GA), referred to as the Divide and Hybridise Algorithm (DaHA), is applied to sequence the DSM with the objectives of minimising iteration and maximising concurrency simultaneously. The DaHA includes a new form of niching, which involves a population being divided into sub-populations, creating an opportunity for each to locate their own local optimum. Sub-populations are then hybridised to explore the solution space between these optima. Further, a new ordinal-based selection method is presented, which encourages diversity and enables a more thorough exploration of the solution space than existing ordinal-based methods. Finally, the DaHA has been compared with several other algorithms from the literature and found to give superior, or at least equal, results when sequencing the DSM.

Keywords: design structure matrix (DSM), genetic algorithm (GA), project management, optimisation

1. Introduction

Optimising product design and development processes is crucial if a business is to be competitive (Karniel and Reich 2009). Structuring these processes requires ordering a large number of activities, to achieve a set goal, in a timely and appropriate manner (Whitfield *et al.* 2003). A classic approach to structuring these processes is to decompose them into sub-systems and analyse how these sub-systems integrate with one another (Browning 2001).

ISSN: 0954-4828 print/ISSN 1466-1837 online
© 2009 Taylor & Francis
DOI: 10.1080/09544820xxxxxxxxxxxxx
http://www.informaworld.com

Steward's Design Structure Matrix[1] (DSM) (Steward 1981a) has been used extensively to represent activities and their dependencies. The DSM provides a concise representation of inter-dependencies for a series of activities. Whilst other methods exist, such as Project Program Evaluation Technique (PERT) and the Critical Path Method (CPM), they are ineffective at analysing the iterative feedback cycles within projects (Steward 1981a, Browning and Eppinger 2002, Meier *et al.* 2007, Browning 2010).

For a process with n activities, there are $n!$ permutations, each representing a possible sequence for the activities to be executed. It is not always possible to simulate each of these sequences in reasonable time frames, particularly if n is large. Thus, the sequencing of a DSM is recognised as an NP-hard optimisation problem (McCulley and Bloebaum 1996, Ahmadi *et al.* 2001, Qian *et al.* 2011). Consequently, in order to sequence the DSM, several meta-heuristic techniques have been used, which seek near-optimal solutions in a reasonable computation time despite the large solution space to be searched. More formally, a meta-heuristic can be defined as a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms (Sörensen and Glover 2013). In this paper a Genetic Algorithm based approach, called the DaHA, has been developed which includes a new form of niching involving a population being divided into sub-populations and crossover being restricted to individuals within the same sub-population. Sub-populations are hybridised to create the next generation of sub-populations containing better offspring and being able to locate their own local optimum which has been shown to achieve superior results to the previously most successful niching method, fitness sharing (Mahfoud 1995, Meier *et al.* 2007). Further, a new ordinal-based selection method has been developed, controlled by a lower limit for selection probability to encourage diversity and enable a more thorough exploration of the solution space, which is shown to outperform existing techniques.

The remainder of the paper is structured as follows. Section 2 introduces the DSM and differentiates between partitioning and sequencing algorithms. Section 3 documents the previous use of Genetic Algorithms to solve DSM problems, then Section 4 provides an overview of the DaHA. Also in Section 4, as mentioned earlier, the new niching method and ordinal-based selection method, which form the main constributions of this work, are presented. In Section 5 the effects of several parameters on the output fitness and run time of the DaHA are investigated. Section 6 analyses the efficacy of the DaHA by comparing its performance with that of existing algorithms reported in the literature. Finally, Section 7 provides concluding remarks.

2. Design Structure Matrix

The DSM provides a visual representation of both the sequence in which activities are executed and the interdependencies among them. Typically, activities are executed in the sequence in which they are listed in the DSM. Sub-diagonal marks represent feed forward information, whereas super-diagonal marks represent feedback information. For example, in Figure 1(a), activities are executed in the sequence A-E. The red cells in the matrix signify instances where information is fed back whereas the blue cells represent instances where information is fed forward.

Browning identified two main categories of DSM: static and time-based (Browning 2001). In time-based DSMs, the sequencing of activities represents a flow through time.

[1]Also known as the Dependency Structure Matrix.

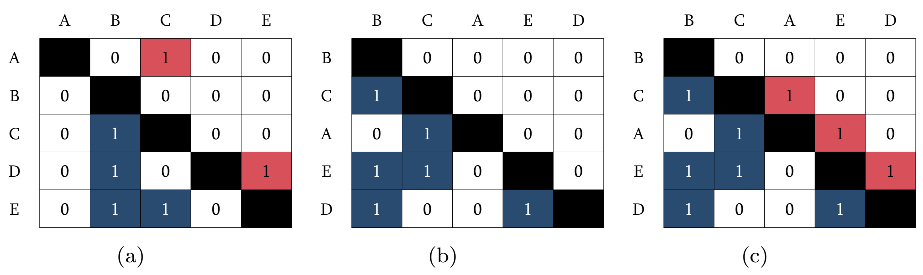


Figure 1.: Simple DSM illustrations.

In static DSMs, the elements all exist simultaneously, such as components of a product or groups of people in a company. For the static DSM, which is not the focus of this paper, the objective is to form modules which contain as few elements as possible and are as independent as possible. Algorithms of this type are called clustering algorithms and there are numerous examples in the literature (Baldwin and Clark 2000, Whitfield *et al.* 2002, Yu *et al.* 2003, 2007, Helmer *et al.* 2010, Borjesson and Hölttä-Otto 2012, Jung and Simpson 2014). The rationale behind increasing modularity is twofold: in component-based DSMs (Browning 2001), it allows the effective use of shared assets to enable cost effective production when companies are offering a variety of products (Otto *et al.* 2013); in team-based DSMs a modular architecture highlights inter-team interfaces which provide the greatest leverage for improving the organisation (Browning 2001).

When sequencing activities in the time-based DSM, the objectives can be to minimise iteration and increase concurrency. The rationale behind having these objectives is that iteration is a major source of increased product development lead-time and cost (Cooper 1993, Rogers 1996, Browning and Eppinger 2002, Meier *et al.* 2007, Karniel and Reich 2009) and by increasing concurrency it is possible to reduce lead-times (Scott 1999). This paper considers only the time-based DSM for which several algorithms exist to re-order their activities.

Algorithms to re-order activities within a DSM have been categorised into two main types: partitioning and sequencing (Meier *et al.* 2007). Whilst both partitioning and sequencing algorithms seek to minimise feedback, they do so in different ways. Partitioning induces a topological sort in an attempt to eliminate feedback marks. Figure 1b shows the same matrix as in Figure 1a after it has been partitioned. Activities that provide information to others have been moved to the start of the DSM. Conversely, activities that require information from others have been moved to the end of the DSM.

Numerous partitioning algorithms exist (Tarjan 1972, Warfield 1973, Steward 1981b) but where iterative loops exist, they can only identify the activities involved, rather than provide a sequence for them (Yassine *et al.* 2000). For example, if more tightly coupled loops are added to the matrix shown in Figure 1b resulting in the matrix shown in Figure 1c, the matrix cannot be fully partitioned, as activities A, C, D and E form an iterative loop: activities A and C both require information from the other activity and so regardless of which is executed first, feedback will always be necessary. The same is true for activities D and E. Iterative loops can also involve more than two activities. For example, of the activities A, C, and E, A requires information from E, E from C, and C from A. It is now unclear how activities A, C, D and E should be sequenced. In real and complex processes, it is unlikely that matrices will fully partition (Yassine *et al.* 2000, Chen *et al.* 2005). For this reason, coupled with the worst case complexity of a partitioning algorithm being $O(n^3)$, Meier at al. have deemed partitioning algorithms as trivial (Meier *et al.* 2007).

Sequencing algorithms are concerned with the ordering of activities within an iterative block (Browning 2001). They are based on the principle of tearing (Steward 1981a,b) whereby assumptions are made about certain dependencies. This allows the process to continue when it reaches a block of coupled activities. Determining the efficacy of a sequence can be measured according to a number of different objective functions, as shown in Table 2 which is presented in Section 6. All objective functions shown in Table 2 seek to minimise the number of tears in the DSM, amongst other things. The reason being that whilst tears allow the process to move forward, they represent assumptions in the process and may lead to iteration (Petz *et al.* 2014). As previously discussed, iteration is a major source of increased product development lead-time and cost. Browning suggested two alternatives to tearing in order to resolve coupled activities in the DSM: aggregation and decomposition (Browning 2001). However, Browning proceeds to mention that aggregation makes the DSM a less useful model in project planning, as it hides the very dependencies it is supposed to expose. Decomposition is the most effective method for resolving coupled blocks in the DSM. However, decomposition is highly problem dependent and is not always possible.

3. Solving DSM problems using Genetic Algorithms

As stated in Section 1, the sequencing of a DSM is recognised as an NP-hard optimisation problem (McCulley and Bloebaum 1996, Ahmadi *et al.* 2001, Qian *et al.* 2011). There are several meta-heuristic techniques which aim to solve this problem including Tabu search (Glover 1989, 1990), Simulated Annealing (Brooks and Morgan 1995) and Ant Colony Optimisation (Dorigo and Birattari 2010). However, particularly for large matrices, Genetic Algorithms (GAs) compare favourably with other optimisation techniques (Meier *et al.* 2007) and are highly prevalent in recent literature in relation to a variety of applications (Roberge *et al.* 2013, Vidal *et al.* 2013, Whitley 2014, Vose 2014, Liu *et al.* 2014). GAs seek to find a good solution to a problem by mimicking natural selection as occurs in nature and are composed of four main operators: objective function evaluation, selection, crossover and mutation (Goldberg and Holland 1988). GAs were first used for scheduling problems by Davis (Davis 1985) and have since been used frequently to sequence the DSM (Altus *et al.* 1996, McCulley and Bloebaum 1996, Rogers 1996, Todd 1997, Scott 1999, Whitfield *et al.* 2003, 2005, Yu *et al.* 2003, 2007, Borjesson and Hölttä-Otto 2012, Jung and Simpson 2014). The remainder of this section provides comments on each of the four operators mentioned in the context of DSM related literature, as well as niching techniques which are a form of restrictive mating within a population.

Depending on the problem, there will be different objectives when sequencing the DSM. Correspondingly, there are several different objective functions used to evaluate the “fitness” of a DSM in the literature. The “fitness” of a DSM is a measure of its effectiveness in solving a particular problem (Meier *et al.* 2007). For the time-based DSM, fitness is a measure of how quickly the process can be executed with the activities in a particular order; whereas for the static DSM, fitness is a measure of how well modules are formed which contain as few elements as possible whilst being as independent as possible (Browning 2001). Due to multiple optimisation criteria for both the time-based and static DSM, several objective functions exist to evaluate fitness. Meier et al. (Meier *et al.* 2007) provided a comprehensive overview of sequencing objective functions, used to evaluate the "fitness" of a DSM, which can be divided into two main groups: those seeking to optimise the time-based DSM (Steward 1981b, Gebala *et al.* 1991, Kusiak and Wang 1993, Altus *et al.* 1996, McCulley and Bloebaum 1996, Rogers 1996, Todd 1997, Scott 1999, Whitfield *et al.* 2005) and those seeking to optimise the static DSM (Baldwin and Clark 2000, Whitfield *et al.* 2002, Yu *et al.* 2003). An overview of the objective functions identified by Meier et al., which have been used to sequence the time-based DSM, can be found in Table 2, which is presented in Section 6.

In terms of selection, there are two main types of method reported in the literature: ordinal-based selection methods such as tournament selection (Brindle 1981) and truncation selection (Mühlenbein 1992), and fitness proportionate methods, such as roulette wheel selection (Goldberg and Holland 1988). The literature on sequencing the DSM is divided between those who use tournament selection (Kusiak and Wang 1993, Altus *et al.* 1996, McCulley and Bloebaum 1996, Rogers 1996) and those who use roulette wheel selection (Todd 1997, Scott 1999, Whitfield *et al.* 2003), with both methods having their strengths and weaknesses. Meier et al. criticise fitness proportionate schemes, stating that the Selection Pressure (SP) is "inadequate" when the difference between fitnesses in the population are either "very high or very low" (Meier *et al.* 2007). Intuitively, one can appreciate that the main problems with tournament selection also occur when there are "very high or very low" differences between fitnesses in the population. For example, if all the solutions had the same fitness, they would not all have the same chance of survival as tournament selection requires solutions to be ranked. When there is no difference between solutions' fitnesses, the ranking is done arbitrarily. Furthermore, if one solution was much fitter than all others, its probability of crossover would be only marginally better than the second best solution's. The worst solution in a population has zero chance of survival in tournament selection, which can lead to a loss of useful genetic information.

Crossover enables a global exploration of the solution space. Briefly, this operator involves two or more parent solutions being chosen by some selection method and then broken down into chromosomes. Next, one or more child solutions are created by taking different chromosomes from the different parents. There are many different crossover operators implemented in GAs. Those that attempt to sequence the DSM frequently use position-based crossover (McCulley and Bloebaum 1996, Rogers 1996, Todd 1997, Scott 1999, Meier *et al.* 2007). It is noted that Syswerda's position-based crossover (Syswerda 1991) is particularly prevalent in the literature (McCulley and Bloebaum 1996, Rogers 1996, Todd 1997, Meier *et al.* 2007).

Whilst crossover is effective at exploring the global solution space, it is ineffective at exploring the local solution space and can lead to irrevocable losses of potentially useful chromosomes (Goldberg and Holland 1988). Unlike crossover, mutation has the ability to create new chromosomes thereby maintaining diversity and possibly yielding better solutions (Meier *et al.* 2007). Typically mutation involves changing one or more genes within a chromosome. There are two principal types of mutation used when sequencing the DSM: order-based mutation (Syswerda 1991) and shift mutation (Murata and Ishibuchi 1994). Whitfield et al. used empirical data to compare the fitness values that different mutation operators produced with results suggesting the superiority of shift mutation (Whitfield *et al.* 2003). De Jong and Brindle showed that high mutation rates will lead a GA to be a random search (De Jong 1975, Brindle 1981). Correspondingly, several authors have recommended a low, fixed mutation probability in the interval $0.001 - 0.01$ (De Jong 1975, Grefenstette 1986).

Niching techniques are used to enhance GAs (Goldberg and Holland 1988) by restricting the likelihood of generating solutions in "crowded regions" of the solution space and thus increase the likelihood of generating solutions in underpopulated regions (Meier *et al.* 2007). To date, there have been two main niching techniques used: crowding (De Jong 1975) and fitness sharing (Holland 1975). Fitness sharing has enjoyed the majority of reported success so far (Mahfoud 1995, Meier *et al.* 2007). However, it is observed that fitness sharing suffers from some of the same problems as fitness proportionate selection. To examine this issue further, it is worthwhile to provide a brief description of fitness sharing. At each generation in a GA, after the objective function has been evaluated for each of the solutions, the solutions are compared against one another to measure how similar they are. Similar solutions will be placed together in a niche. At this stage, the niches are treated exactly like individual solutions in fitness proportionate selection: they are provided with a probability of crossover which is proportionate to their fitness. This probability of crossover is then divided by the number of solutions in that niche. A problem with this method is exactly the same as Meier et al. identified in fitness proportionate selection (Meier *et al.* 2007): Selection Pressure (SP) will be inadequate when the differences between niches are either "very high or very low".

4. Divide and Hybridise Algorithm (DaHA)

An overview of the enhanced GA, referred to as the DaHA, is presented in this section. The DaHA includes novel contributions in two areas: niching and selection. In terms of niching, the population is divided into a number of sub-populations and crossover is restricted to individuals within the same sub-population enabling each sub-population to locate its own local optimum. Further, in an iterative manner, the current set of sub-populations are then hybridised to create the next set of sub-populations with the aim of exploring the solution space between the previously located local optima. Unlike in fitness sharing, a sub-population, which effectively becomes a niche, has a fixed size regardless of its fitness, thus eradicating the problem of inadequate selection pressure (SP) when the differences between niches are either "very high or very low". For selection, an ordinal-based method is employed, referred to as the multiplier-driven selection method. This method uses a multiplier, which can vary continuously, to provide each solution in a sub-population with a rank-based selection probability to encourage fitter individuals to be selected as parents thus increasing the likelihood of fitter offspring. Also, a lower limit for selection probability is employed to allow even the weakest solutions to be selected as parents, with the aim of encouraging diversity and enabling a more thorough exploration of the solution space.

4.1. *Overview of the DaHA*

The DaHA is aimed at sequencing the DSM with the objective of minimising iteration and maximising concurrency simultaneously. Pseudocode of the DaHA is presented in Algorithm **1**.

[1]Objective function.

Algorithm 1 DaHA

```
1: hybridisations = 0
2: MAX = 50
3: $S \leftarrow$ sub-population
4: $x \leftarrow$ sub-population size
5: $y \leftarrow$ number of sub-populations        ▷ $S[i]$ denotes $i^{th}$ solution in a sub-population
6:                                               ▷ $S_i$ denotes $i^{th}$ sub-population
7: $P \leftarrow$ INITIALISE (popSize)                ▷ Initialise Population $(xy)$
8: while NotConverged($S_1[1], S_2[1]...S_y[1]$) and hybridisations $\leq$ MAX do
9:     for $i \leftarrow 1$ to $y$ do
10:        Apply GA to $(S_i)$   ▷ OF[1] evaluation, elitism, selection, crossover, mutation
11:    for $i \leftarrow 1$ to $y$ do
12:        for $j \leftarrow 1$ to $x$ do
13:            if j is divisible by 2 then
14:                if $j = 2$ then
15:                    $S_i[j] = S_{i+1}[1]$          ▷ Takes the best solution from $S_{i+1}$
16:                else $S_i[j] = S_{i+1}[j]$
17:            else $S_i[j] = S_i[j]$
18:    hybridisations++;
19: Local search around best solution found
```

Line 10 of Algorithm **1** refers to five operations performed on each sub-population considered, namely objective function evaluation, elitism, selection, crossover and mutation. In terms of the objective function to be evaluated, the DaHA has the objective of minimising iteration and maximising concurrency simultaneously according to Equation 1, where F is the defined fitness (to be minimised), n is equal to the number of activities in the DSM, $w(i,j)$ refers to the cell value on the i^{th} row and j^{th} column of the DSM, $\Omega(i,j) = [j + (n-i)]^2$ if $i > j$ and $\Omega(i,j) = 100.[j + (n-i)]^2$ if $i < j$ (Scott 1999).

$$F = \sum_{i=1}^{n} \sum_{j=1}^{n} \Omega(i,j).w(i,j) \tag{1}$$

Importantly, in the DaHA, the objective function to be evaluated can be replaced with that used by other sequencing algorithms thus allowing a direct comparison of performance to be made, which is discussed in Section 6.

Further, elitism was implemented such that the top k% of solutions were included, unaltered, in the next generation[1]. In terms of crossover and mutation, Syswerda's position-based crossover (Syswerda 1991) was implemented owing to its prevalence in the literature as discussed in Section 3 and shift mutation was used following its suggested superiority to order-based mutation (Whitfield *et al.* 2003, Meier *et al.* 2007).

In the DaHA, two termination criteria are used: (1) if the number of hybridisations exceeds a user defined maximum; (2) if the best solutions from each of the sub-populations $S_1[1]$ to $S_y[1]$ have converged. Both of these termination criteria are nested inside the *while* loop of Algorithm **1** (line 8). Based on a preliminary investigation, the maximum number of hybridisations in the DaHA was set to 50 since if the population had not converged after 50 hybridisations, it often did not converge at all. To determine whether or not the best solutions from each of the y sub-populations, $S_1[1]$ to $S_y[1]$, had converged,

[1]The optimum value for k for a given DSM is investigated in Section 5.

a convergence ratio (CR) was used, as defined by McCulley and Bloebaum (McCulley and Bloebaum 1996). This convergence ratio is defined in Equation 2, where x is the size of the sub-population, $F[i]$ is the fitness value for the i^{th} solution in the sub-population and $F[1]$ is the fitness value for the best solution in the sub-population, as defined by the objective function. If the convergence ratio of the population was greater than a user defined threshold (CT), which was set between 0 and 1, the GA terminated. The effect of this convergence threshold on fitness and run time is investigated in Section 5.

$$CR = \frac{x.F_1}{\sum\limits_{i=1}^{x} F_i} \tag{2}$$

4.2. *Niching*

As referred to in Section 3, niching techniques are used to restrict the likelihood of generating solutions in "crowded regions" of the solution space and thus increasing the likelihood of generating solutions in underpopulated regions (Meier *et al.* 2007). Typically, as a GA converges on a solution, the population becomes more and more homogeneous, thus "crowding" around one area of the solution space. With reference to Algorithm **1**, by dividing the population into y sub-populations (line 5) and applying GA operations on each of the sub-populations separately (line 10), the DaHA leads the total population to "crowd" around y different areas of the solution space, rather than just one. The reason for this is that GAs have a random element and will likely converge on several different solutions if replicated multiple times[1]. By limiting the likelihood of generating solutions in already "crowded regions" of the solution space, the DaHA acts as a niching method. However, if the sub-populations were kept separate throughout the application of Algorithm **1**, then the effect would be equivalent to employing a traditional GA y times with a population size (P/y), where P is the total population size and y is the number of sub-populations.

The specifics of this hybridisation method, illustrated in Algorithm **1** (lines 8-18), are as follows: a new, child sub-population, S_i^*, is derived from two parent sub-populations, S_i and S_{i+1}, by taking the best solution from each parent sub-population, $S_i^*[1] = S_i[1]$ and $S_i^*[2] = S_{i+1}[1]$ and then taking alternating solutions from each parent sub-population: $S_i^*[3] = S_i[3]$, $S_i^*[4] = S_{i+1}[4]$ and so on (lines 13-17). Keeping sub-populations separate allows each sub-population to approach its own local optimum when a GA is applied (line 10). The iterative hybridisation of sub-populations, each time all sub-populations have converged, allows the exploration of the solution space between these local optima.

4.3. *Selection*

Selection techniques provide a probability of crossover for each solution in the sub-population. In fitness proportionate schemes, the probability of crossover is found according to Equation 3, where $Q[i]$ is the probability of crossover for the i^{th} solution, x is the size of the sub-population and $F[i]$ is the fitness value for the i^{th} solution in the sub-population according to the objective function.

[1]This effect is especially pronounced when there is insufficient population (see Section 5).

$$Q[i] = \frac{F[i]}{\sum\limits_{i=1}^{x} F[i]} \tag{3}$$

In tournament selection methods, t solutions are chosen at random and their fitnesses are compared with the solution having the highest fitness being selected. In effect, this provides each solution with a probability of crossover according to Equation 4[3].

$$Q[i] = \frac{(x-i)^{t-1}}{\sum\limits_{i=1}^{x} (i-1)^{t-1}} \tag{4}$$

In contrast to tournament selection methods, in which the parameter t can only vary in discrete steps, the DaHA employs the multiplier-driven selection method which uses a multiplier M, ranging from 0 to a user defined upper value, that can vary continuously. Consequently, Equation 4 can be modified to give Equation 5.

$$Q[i] = \frac{(x-i)^{M}}{\sum\limits_{i=1}^{x} (i-1)^{M}} \tag{5}$$

Figure 2 shows how M affects the population's probability of selection. Tournament selection with two competitors (Altus *et al.* 1996, McCulley and Bloebaum 1996, Rogers 1996) and four competitors (Meier *et al.* 2007) are equivalent to a probability distribution when the multiplier, M, is equal to 1 and 3 respectively. The ability to vary the multiplier M continuously provides a selection method more flexible than traditional tournament selection. However, the use of the multiplier M does require a population's solutions to be sorted; a process of $xlog(x)$ complexity where x is sub-population size. The multiplier's effect on fitness and run time is explored in Section 5.

In the multiplier-driven selection method, a lower limit on a solution's probability of crossover was introduced to ensure even the weakest solutions have a probability of being selected as parents. The use of the lower limit provides the potential to avoid the loss of useful genetic information, which occurs in tournament selection methods where the worst solution has zero chance of survival even if only marginally worse than the best solution. Historically, the loss of useful genetic information in a GA has been managed by using mutation to introduce new chromosomes into the gene pool. However, it is observed that mutation is inherently random in nature, and if used too often in trying to maintain diversity, can lead a GA to become a random search (De Jong 1975, Brindle 1981). Contrastingly, the lower limit can be used to maintain diversity whilst avoiding the GA becoming a random search. The population's crossover probabilities, calculated using Equation 5, were altered according to Equation 6 in which a user defined lower limit (L), that could vary between 0 and 1, was multiplied with the highest probability of crossover in the sub-population Q[1]. This scaled lower limit is then added to all solutions probability of selection.

[3]Where solutions have been sorted such that the best solution has $i = 1$ and the worst solution has $i = x$.

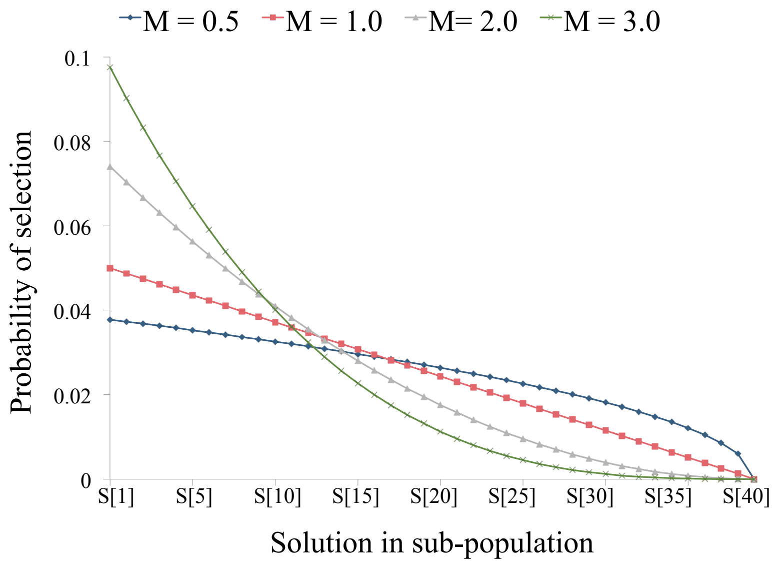


Figure 2.: Probability of selection for different multipliers.

$$Q[i] = \frac{Q[i] + Q[1].L}{1 + x.Q[1].L} \tag{6}$$

Algorithm **2** presents the multiplier-driven selection method. Each sub-population is initially sorted such that $S[1]$ is the best solution and $S[x]$ is the worst solution (line 9). An initial, ordinal-based probability of selection is then calculated (line 11) based on a solution's position in the order of sorted solutions. Next, for each solution, a range $T[i]-B[i]$ is calculated (lines 13-17) by adding a scaled lower limit calculated in line 12 to the ordinal-based probability of selection calculated in line 11. This range $T[i] - B[i]$ has a minimum of 0 and maximum of 1, with the size determining the solution's probability of selection. The function of the denominator in line 17 and in Equation 6, is to normalise the probabilities of selection so that they sum to unity.

An illustration of the effect of the lower limit on probability of selection is shown in Figure 3. Each of the data points located on the red line have been altered in accordance with Equation 6 to produce the blue line. The effect of the lower limit on fitness and run time is explored in Section 5.

5. Tuning the performance of the DaHA

Before applying the DaHA to the same DSM problems as a number of other algorithms, an investigation was undertaken to determine the 'optimised' settings of GA-based parameters, which would yield the best performance of the DaHA in terms of solution fitness and run time. The motivation for doing so was to enable a fair comparison between the application of the DaHA with other DSM sequencing algorithms, as presented in Section 6, given these will themselves will have been tuned. Furthermore, prior to this investigation, a sensitivity analysis, not presented in this paper, was carried out to establish the order in which these parameters should be investigated, given the varying impact

Algorithm 2 Selection process

```
1:  $S \leftarrow$ sub-population
2:  $x \leftarrow$ sub-population size        ▷ $S[i]$ denotes $i^{th}$ solution in a sub-population
3:  $M \leftarrow$ selection multiplier
4:  $L \leftarrow$ selection lower limit
5:  $Q \leftarrow$ ordinal-based probability of selection
6:  $B \leftarrow$ bottom of solution's selection range
7:  $T \leftarrow$ top of solution's selection range
8:  $a, b, c, d, e \leftarrow$ dummy variables
9:  Sort(S)        ▷ Such that $S[1]$ is the best solution and $S[x]$ is the worst
10: for $i \leftarrow 1$ to $x$ do
11:     $Q[i] = (x - i)^M$
12: $a = Q[1].L$        ▷ Define a scaled lower bound for probability of selection
13: for $i \leftarrow 1$ to $x$ do
14:     if $i = 1$ then
15:         $B[i] = 0$
16:     else $B[i] = T[i - 1]$
17:     $T[i] = B[i] + ((Q[i] + a)/(1 + xa))$
18: Create two random numbers b and c        ▷ In the range $0 < b, c \leq 1$
19: for $d \leftarrow 1$ to $x$ do
20:     if $B[d] < b \leq T[d]$ then return
21: for $e \leftarrow 1$ to $x$ do
22:     if $B[e] < c \leq T[e]$ then return
23: Parents are $S[d]$ and $S[e]$
```

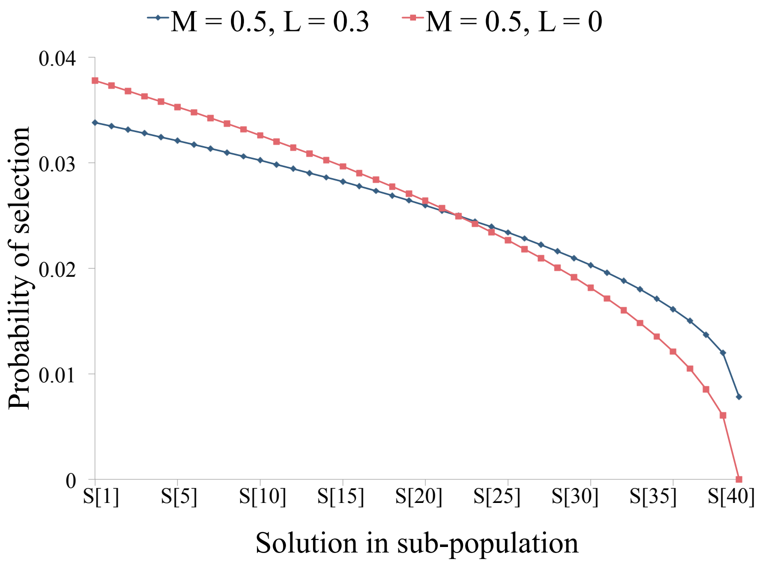


Figure 3.: Probability of selection with lower limit.

of each on the fitness of solutions obtained. The order of investigation established, from most to least impact on solution fitness, was as follows: population size, selection and elitism parameters, convergence threshold and mutation probability. For each parameter setting to be determined, 50 experimental runs of the DaHA were replicated. Also, the objective function used in all experimental runs is as defined in Equation 1 stated in Section 4.1. The CPU used was an Octal Core, 3.50 GHz Intel Core i7-3770K with 16GB RAM.

5.1. *Effect of population size*

Population size has been recognised as having a pronounced effect on the efficacy of a GA (Goldberg and Holland 1988). For DSM problems, determining a sufficient population size to provide a good solution, without unnecessary computational expense, has been discussed in the literature with different authors providing various methods to determine this size (McCulley and Bloebaum 1996, Scott 1999, Meier *et al.* 2007). In this paper, McCulley and Bloebaum's definition of sufficient population is adhered to: "sufficient population" is "the first population size for which the system converges to within 5% of the maximum fitness".

With regard to the DaHA, population size is defined as the product of sub-population size and the number of sub-populations. In this paper, sufficient population size was determined for a series of fifty different DSMs varying in both number of activities represented, i.e. matrix size, and number of dependencies between these activities. However, in order to perform a complete search, determining simultaneously the optimum sub-population size and number of sub-populations for each of the fifty DSMs considered would have incurred unacceptable computation times. Thus, an estimate was made on sub-population size so that only the number of sub-populations had to be varied to determine sufficient population. A sub-population size of 40 was chosen initially on the basis that it would provide an appropriate degree of diversity within a sub-population and allow solutions to be obtained in reasonable computation time. Further, the use of a sub-population size of 40 would be revisited, and revised if appropriate, once the sufficient population size had been determined.

When determining sufficient population in a GA to sequence the DSM, both matrix size and coupling density have been shown to have a significant effect (McCulley and Bloebaum 1996). The coupling density (CD) of a DSM, defined in Equation 7[1], is a function of the number of activities represented in the matrix, z, and the number of dependencies between these activities, d.

$$CD = \frac{d}{z(z-1)} \tag{7}$$

For a number of binary DSMs considered in the literature, Table 1 lists the corresponding matrix size, number of dependencies and coupling density. Based on the data shown in Table 1, it was decided to investigate the sufficient population for fifty DSMs with sizes $10 - 100$ in intervals of 10 and coupling densities $0.03 - 0.15$ in intervals of 0.03. As sub-population size had been set initially to 40, the number of sub-populations in the DaHA was then varied when sequencing each of the matrices to determine sufficient population.

Figure 4 shows the effect of matrix size (z) and coupling density (CD) on the number of sub-populations required to give a sufficient population. As the size of the DSM increases,

[1]Self-dependencies, which are located along the diagonal of the DSM, are not included.

Author(s)	Size of DSM (z)	Number of dependencies (d)	Coupling Density (CD)
Steward 1981b	20	45	0.1184
Gebala *et al.* 1991	37	141	0.1059
Todd 1997	51	203	0.0796
Scott 1999	60	147	0.0415
McCulley and Bloebaum 1996	42	146	0.0848
Eppinger *et al.* 2013	92	1098	0.1312
Mean	50	297	0.0936

Table 1.: Coupling densities.

the problem becomes more complex thus requiring larger population sizes. Furthermore, as McCulley and Bloebaum observed (McCulley and Bloebaum 1996), and Meier et al. have since explored (Meier *et al.* 2007), lower coupling densities require larger population sizes. As shown in Figure 4, this effect is particularly evident on the matrices with a coupling density of 0.03. The sufficient population sizes are up to 100% greater than for all other matrices. Furthermore, the DSMs with $CD = 0.03$ exhibit less of a trend line for the number of sufficient populations versus matrix size.

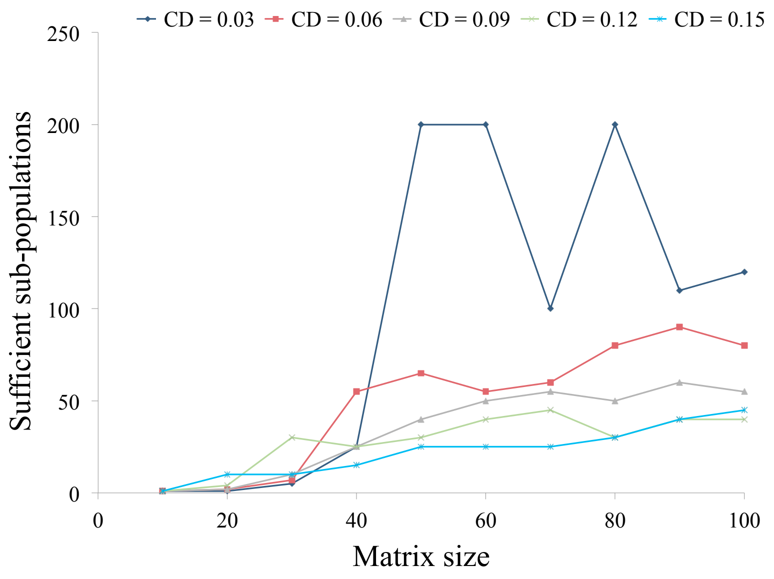


Figure 4.: Effect of DSM size and coupling density on number of sub-populations.

Thus far, the results presented were obtained using fifty different DSMs. Hereafter, this paper considers only one matrix with both mean size and coupling density according to Table 1, i.e. a 50×50 matrix with $CD = 0.09$. With this matrix size and coupling density, the initial estimate of sub-population size was tested to ensure that 40 was a suitable value. Figure 4 shows that for a 50×50 matrix with $CD = 0.09$, the sufficient number of sub-populations was also 40, and with a sub-population size of 40 this gives a total population size of 1600. With the total population size set at 1600, the effect of varying sub-population size and the number of sub-populations, on solution fitness and run time was investigated. As sub-population size was increased, the number of sub-populations was decreased to maintain the total population size of 1600. Figure 5 shows that for a fixed population size of 1600, sub-population size has a minimal effect on fitness, with all solutions achieving within 1% fitness of each other. Note that in Figure 5, and subsequently in Figures 6-10, absolute fitness values are used to enable the relative effects of parameters to be compared directly. These fitness values were found using the objective function defined in Equation 1 and then dividing by the fitness value of the best solution found for the 50 × 50 matrix with $CD = 0.09$.

Also shown in Figure 5, sub-population size can be seen to have a major effect on run time. Run times for a sub-population size of 15 were more than 5 times greater than at a sub-population size of 20. This is due to sub-populations taking longer to converge (line 8 in Algorithm **1**). Furthermore, there is a steady increase in run time for sub-population sizes greater than 40. This is attributed to the GA taking longer to converge for each of the sub-populations (line 10 of Algorithm **1**). In consideration of the results shown in Figure 5, it was concluded to keep both sub-population size and the number of sub-populations at 40.

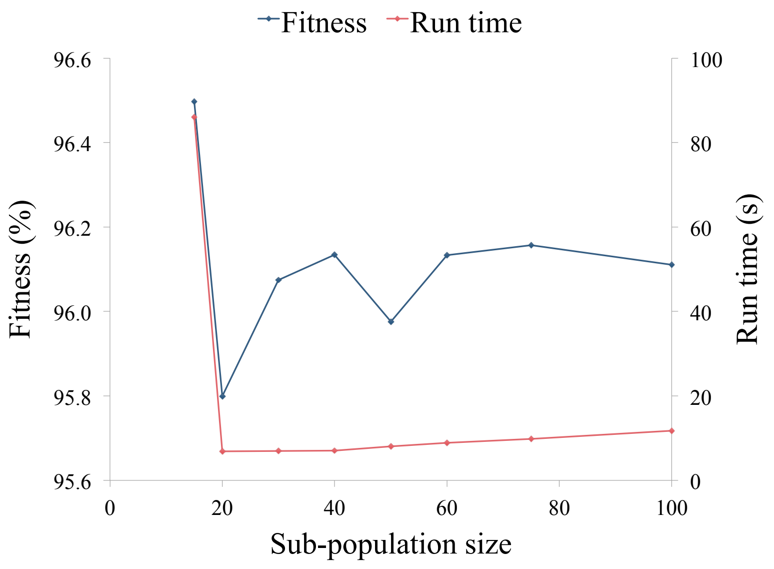


Figure 5.: Sub-population size versus fitness and run time.

5.2. *Effect of selection parameters*

As discussed in Section 4, the DaHA has two selection parameters, both of which can vary continuously: the selection multiplier (M) and the lower limit (L).

In the DaHA, the multiplier M provides selection pressure (SP). Meier et al. stated that "when SP is low, a genetic drift occurs causing the GA to converge arbitrarily to a solution" and "SP that is too high results in premature convergence" (Meier *et al.* 2007). Figure 2 in Section 4.3 shows how four sample multipliers (M=0.5, 1.0, 2.0, 3.0) affect a population's probability of selection. The effect of the selection multiplier on fitness and run time can be seen in Figure 6, which shows that a lower multiplier, such as 0.5, can provide better solutions with only a 10 second increase in run time compared to higher selection pressures used in (Kusiak and Wang 1993, McCulley and Bloebaum 1996, Rogers 1996, Meier *et al.* 2007).

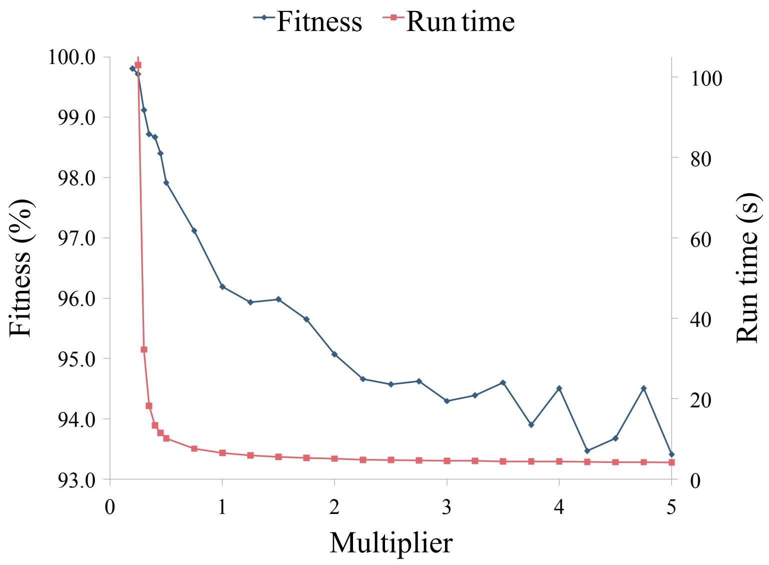


Figure 6.: Selection multiplier versus fitness and run time.

The lower limit L was varied in the DaHA to give the results presented in Figure 7, which shows that increasing the lower limit to leads to increased fitness and run time. However, it can also be seen that both fitness and run time are reduced as the lower limit approaches unity, which is due to a change in the termination criteria used. Specifically, the DaHA changes from terminating due to convergence to terminating due to reaching the maximum number of hybridisations in the DaHA (line 8 in Algorithm **1**).

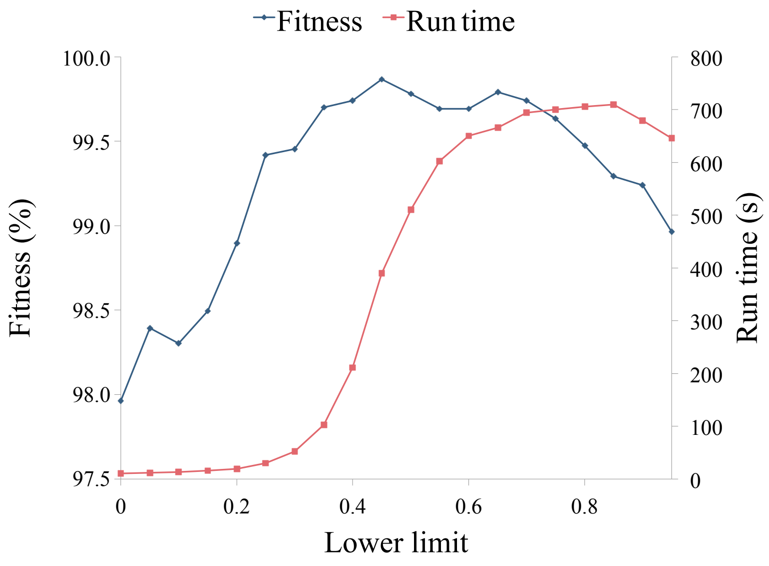


Figure 7.: Selection lower limit versus fitness and run time.

In summary, lower selection pressure can lead to better solutions at the expense of increased run times. Further, when the values of these selection parameters are chosen carefully, such as a multiplier M=0.5 (see Figure 2) and a lower limit L=0.2 (see Figure 7), results indicate that it is possible to achieve within 1.5% of the best achieved fitness with run times of less than 60 seconds. Thus, a multiplier of 0.5 and a lower limit of 0.2 were used hereafter in the DaHA.

5.3. *Effect of elitism parameter*

Elitism was implemented in the GA such that a proportion of the fittest individuals within a sub-population were included, unaltered, in the next generation. For example, an elitism parameter value of 0.15 corresponds to the fittest 15% of solutions, as defined by the objective function, being included, unaltered, in the next generation. In the DaHA, this parameter was varied between 0 and 1.0 to give the results presented in Figure 8, which shows the effect of elitism on fitness and run time. Empirical results have shown previously that elitism speeds up convergence and can also lead to increased fitness values (Laumanns *et al.* 2000) since solutions in the elitist population provide guidance towards local optima. However, too much elitism can cause the GA to converge prematurely.

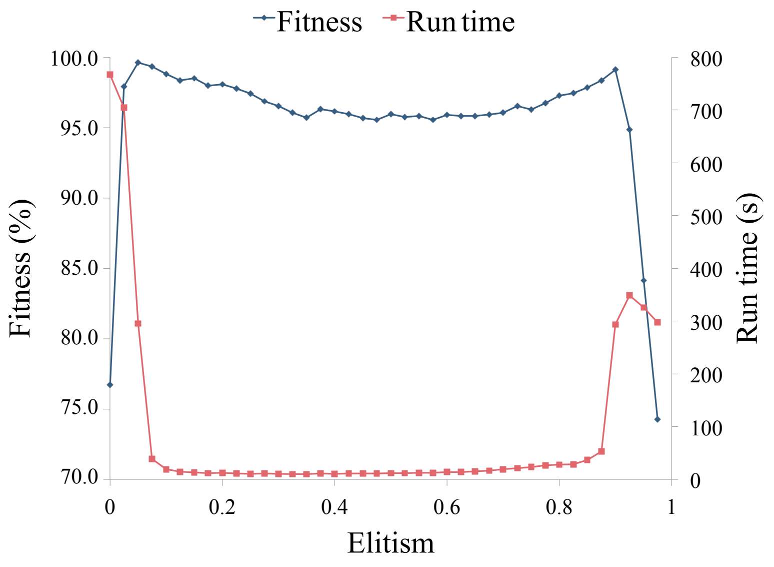


Figure 8.: Elitism versus fitness and run time.

Figure 8 shows that in the lower region of the elitism range, a peak fitness value of 99.6%, with an associated run time of 296s, is obtained with an elitism parameter value of 0.05. In the upper region of the elitism range, a peak fitness value of 99.1%, with an associated run time of 293s, is obtained with a value of 0.9. Between these elitism parameter values, fitness remains within 5% of the maximum obtained, however predominantly the run times are significantly less. Elitism parameter values less than 0.05 or greater than 0.9 lead to significant reductions in fitness. For parameter values less than 0.05 this is attributable to solutions receiving little guidance towards local optima, also leading to significantly increased run times. For parameter values greater than 0.9, the reduction in fitness and relatively shorter run times are attributable to a change in which of the termination criteria are applied. That is, as in Section 5.2, the DaHA changes from terminating due to convergence to terminating due to reaching the maximum number of hybridisations in the DaHA. By inspection of Figure 8, it is observed that choosing an elitism parameter value of 0.1 can lead to fitnesses greater than 99% of the best solution found in run times less than 60 seconds. Thus, the elitism parameter was set as 0.1 hereafter in the DaHA.

5.4. *Effect of convergence threshold*

As discussed in Section 4.1, the GA implemented in line 10 of Algorithm **1** terminated when convergence, as defined in Equation 2, exceeded a user defined convergence threshold. Figure 9 shows that a convergence threshold below 0.9 has no obvious effect on the DaHA's performance in terms of solution fitness and run time. However, when the convergence threshold exceeds 0.9, run times are consistently less than when it is lower than 0.9. Further, a steady increase in fitness can be observed as the convergence threshold rises. To maximise solution fitness, the convergence threshold was set to 0.99 hereafter in the DaHA.

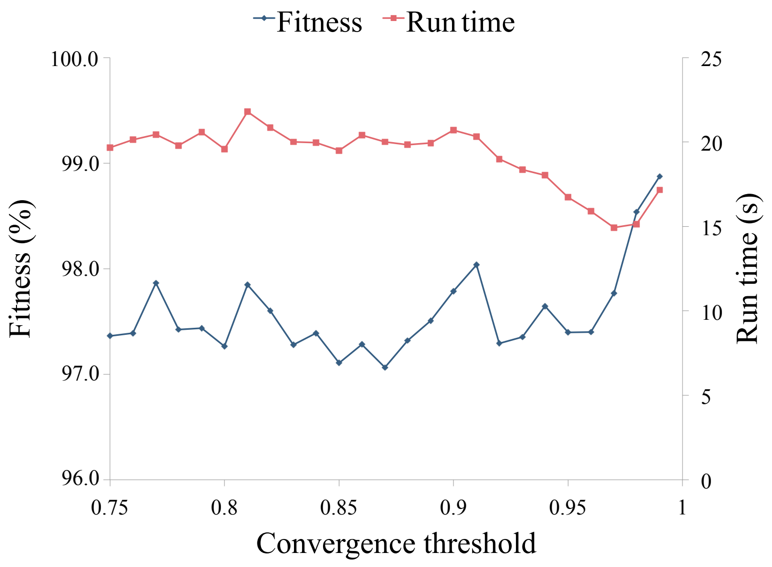


Figure 9.: Convergence threshold versus fitness, run time, generations to converge.

5.5. *Effect of mutation probability*

Figure 10 shows that higher probabilities of mutation lead to increased fitness at the expense of longer run times, which is due to mutation introducing new genetic information into the gene pool and thus creating a more varied population. More specifically, Figure 10 shows that run times increase quadratically with respect to mutation probability. Conversely, the benefit of mutation probability on fitness appears to decrease quadratically. Given a mutation rate of 0.1 resulted in solutions within 1% of the peak fitness in run times of under 20 seconds, the mutation rate in the DaHA was set at this value.

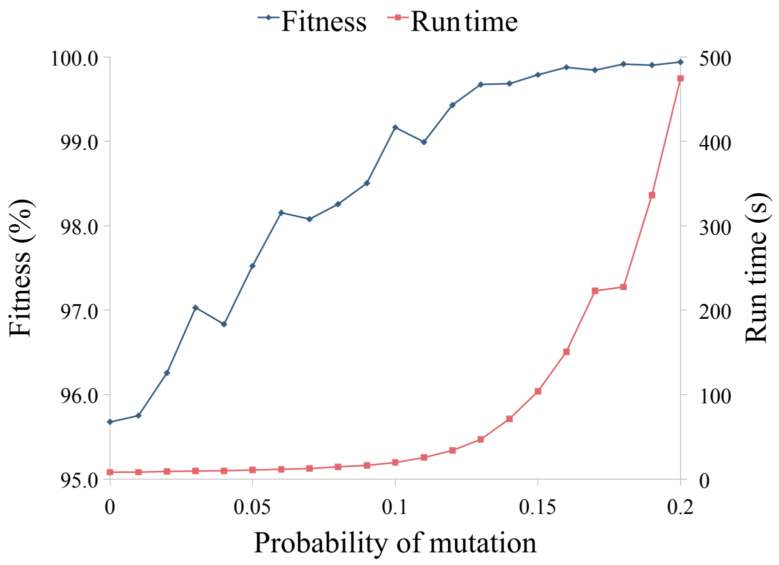


Figure 10.: Mutation probability versus fitness and run time.

6. Comparison of the DaHA with other DSM sequencing algorithms

Following the performance tuning of the DaHA, it was applied to a series of sequencing problems of the time-based DSM, presented in the literature, as identified by Meier et al. (Meier *et al.* 2007). As mentioned in Section 2, the DSM is context dependent. Whilst the DaHAs parameters were only tuned for a specific DSM, this DSM was constructed by taking an average size and coupling density of DSMs used in several case studies as discussed in Section 5. The results presented in this section illustrate that further tuning of the DaHA is not necessary to achieve equal or better solutions to other authors sequencing algorithms despite using different objective functions to compete with each. This demonstrates the versatility of the DaHA to sequence the DSM. For each problem considered, the DaHA sought to optimise the DSM using the same objective function as the author(s) of the respective paper. In each case, the author's or authors' original DSM was resequenced into a random sequence before the DaHA was applied to it. Of the time-based DSM problems identified by Meier et al. (Meier *et al.* 2007), some papers were excluded from our comparison analysis for various reasons. For example, neither Altus (Altus *et al.* 1996) or Rogers (Rogers 1996) were included because their respective objective functions are problem dependent. Whitfield et al.(Whitfield *et al.* 2005) was not included since the objective function does not provide an equation to evaluate the fitness of the DSM. Finally, Kusiak (Kusiak and Wang 1993) was neglected since for binary DSMs the objective function is equivalent to Steward's objective function (Steward 1981b). Table 2 presents a summary of one DSM problem from each of the remaining papers identified by Meier et al. (Meier *et al.* 2007).

Table 2 shows that the use of the DaHA led to an improvement for all DSM problems considered, except Scott's (Scott 1999). In this table, the column headed *Original solution* refers to the solution value of the respective author's or authors' optimised sequenced DSM using their objective function. Note that the use of different objective functions by different authors, coupled with considering different DSM problems, explains why the solution values stated vary. The column headed *DaHA solution* in Table 2 refers to the solution values obtained using the DaHA, which are lower (better) than those

Author(s)	Objective	Rationale	Objective Function[a]	Original solution	DaHA solution	Improvement (%)
Without meta-heuristic optimisation						
Steward 1981b	Minimise number of feedback marks	Reduce iteration	$\sum_{i=1}^{n}\sum_{j=1}^{n} w(i,j)$	18	5	72.2
Gebala *et al.* 1991	Minimise total feedback length; i.e., Minimise distance from diagonal	Reduce iteration length	$\sum_{i=1}^{n}\sum_{j=i+1}^{n} (j-i).w(i,j)$	97	48	50.5
With meta-heuristic optimisation						
Todd 1997	Push marks to the left (C0) and bottom (C1) of DSM, minimise total feedback length (C2)	Jointly minimise iteration and improve concur-rency	Criterion 0: $\sum_{i=1}^{n}\sum_{j=1}^{n} j.w(i,j)$ Criterion 1: $\sum_{i=1}^{n}\sum_{j=1}^{n} (n-i).w(i,j)$ Criterion 2: $\sum_{i=1}^{n}\sum_{j=i+1}^{n} (j-i).w(i,j)$	C0: 3865 C1: 4752 C2: 174	C0: 3810 C1: 4748 C2: 172	C0: 1.42 C1: 0.08 C2: 1.15
Scott 1999	Jointly reduce feedback marks and move marks to bottom left-hand corner of DSM	Jointly minimise iteration and improve concur-rency	$\sum_{i=1}^{n}\sum_{j=1}^{n} \Omega(i,j).w(i,j)$ where $\Omega(i,j) = [j+(n-i)]^2$ if $i > j$ and $\Omega(i,j) = 100.[j+(n-i)]^2$ if $i < j$	2,456,672[b]	2,456,672[c]	0.0
McCulley and Bloebaum 1996	Minimise feedback and crossovers	Reduce number of activities involved in an iteration loop	$w_f.f + w_c.c$ where f and c are the number of feedbacks and crossovers respectively and w_f and w_c are the associated user-defined weights.	32.8[d]	28.2[d]	14.0

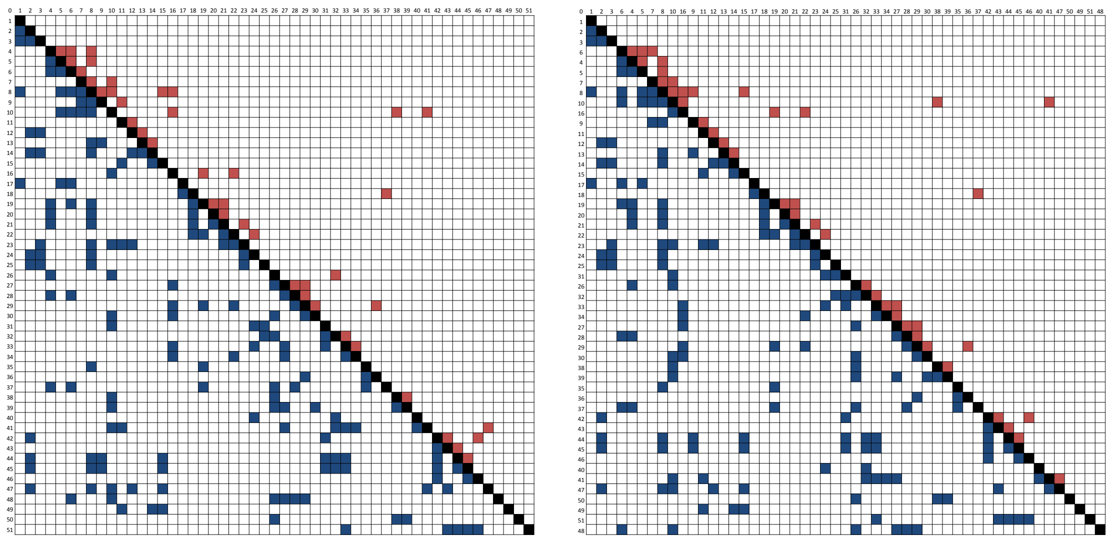
a) On replicating the work of Scott, by multiplying the binary value in each cell of his DSM with the weighted distance value in the corresponding matrix position to obtain a measure of inter-activity iteration, we found that Scott's actual solution was better than that which he claimed. That is, Scott's claimed result of 2,757,320 should be 2,456,672.
b)Using a different ordering of the DSM.
c)Where $w_f = 0.9$, $w_c = 0.1$.

Table 2.: Comparison of the DaHA with other sequencing algorithms (adapted from (Meier *et al.* 2007)).

obtained by the respective authors, again with the exception of Scott's solution. These lower values obtained using the DaHA indicate the algorithms usefulness and versatility in that it is able to achieve better solutions in all cases except one when applied to the same DSM problem and using the same objective function of each author(s). It is noteworthy that neither Steward (Steward 1981b) or Gebala and Eppinger (Gebala *et al.* 1991) used meta-heuristic techniques to optimise the sequence of their DSMs. This is the likely reason that the DaHA, with its ability to search large solution spaces and find a good or near-optimal solution in a reasonable amount of time, was able to make such significant improvements on their solutions, approximately 72% and 50% respectively. Thus, it is considered that this is not an appropriate comparison to make, although a visual comparison of the solutions obtained using the DaHA and those reported by Steward (Steward 1981b) and Gebala and Eppinger (Gebala *et al.* 1991) are presented in Appendix A.

GAs were used by Todd (Todd 1997), Scott (Scott 1999), and McCulley and Bloebaum (McCulley and Bloebaum 1996) to sequence their respective DSMs and thus their algorithms are suitable for comparison with the DaHA. It is noted that the greatest improvement achieved was in the comparison with McCulley and Bloebaums solution. A suggested reason for this could be that McCulley and Bloebaums DSM has the closest coupling density to that for which the parameters of the DaHA were tuned in Section 5. Thus, one might expect the DaHA to make larger percentage improvements on this matrix compared to others.

For the DSM problem considered by Todd (Todd 1997), weightings were not stated for any of the three objective function criteria used. However, by assuming these weightings, the DaHA was able to be applied to the same DSM problem resulting in several solutions that improved on Todd's solution for all three criteria simultaneously, as indicated in Table 2. The assumed weightings W0, W1, W2 for the corresponding three criteria were calculated by multiplying the reciprocal of Todd's solution for that criterion (e.g. 1/3810 for C0) by an empirically found constant E1, E2 and E3 respectively. The empirical constants were found by applying the DaHA repeatedly, using trial and improvement, to obtain solutions that (1) improved on Todd's DSM solution for all three criteria and (2) provided the largest possible sum percentage improvement for the three criteria. Figures 11a and 11b present the DSM solutions obtained by Todd and the DaHA respectively. The upper half of the DSM (up to activity 25) is largely the same in both Todd's and the DaHA's solution. The only differences are that activities 4, 5 and 6 have been resequenced by the DaHA in the order 6, 4 and 5 and activities 9 and 10 have switched order with activity 16 now appearing in between them. However, the lower half of the DSM, following activity 25, has undergone a major resequencing. This is most apparent by the differences in the bottom right quadrants of Figures 11a and 11b. As the percentage improvements for C0, C1 and C2 were all less than 1.5%, Figures 11a and 11b are ineffective in showing how dependencies in the DSM have been moved to the bottom left corner and how feedback length has been minimised.



(a) (b)

Figure 11.: (a) Todd's solution (Todd 1997) and (b) DaHA's solution.

For Scott's (Scott 1999) DSM problem and associated objective function, multiple experimental runs were carried out, however the DaHA always converged on either the same sequence obtained by the author, or another sequence with the same solution fitness. Table 2 shows that Scott's objective function weights feedback information as 100 times more important than feed forward information. For this reason, unless the DaHA was able to reduce the number of feedback marks in Scott's DSM, it was unlikely to improve on the result. With Scott's sequenced DSM, shown in Figure 12a, containing only five feedback marks out of 147 dependencies, the DaHA was unable to improve on this result. The DSM obtained by applying the DaHA is shown in Figure 12b. Whilst the two DSM solutions appear to be almost identical, on closer inspection one can see that activities 20, 21 and 22 are ordered differently in the DaHA's solution but result in exactly the same distribution of feedback and feed forward marks owing to similar dependencies of the activities. The same is true of activities 54 and 55.

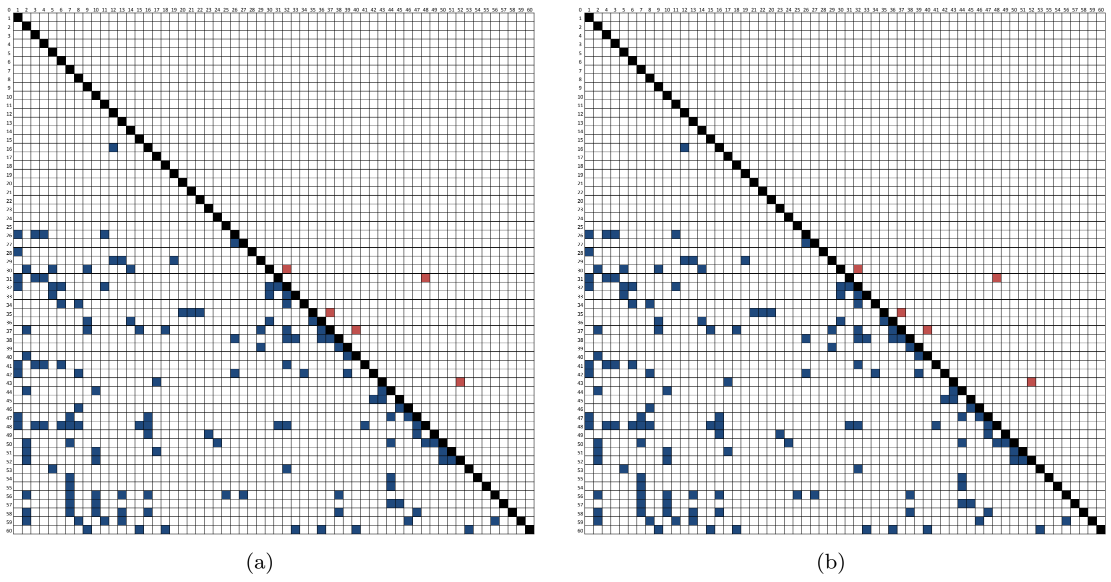


Figure 12.: (a) Scott's DSM solution (Scott 1999) and (b) DaHA's DSM solution.

McCulley and Bloebaum's DSM (McCulley and Bloebaum 1996) was re-sequenced by the DaHA to provide the largest percentage improvement for a problem that had been sequenced originally using meta-heuristic techniques. Figures 13a and 13b present the DSM solutions obtained by McCulley and Bloebaum and the DaHA respectively. The solution found by the DaHA sequences the vast majority of activities in a different order to McCulley and Bloebaum. Consequently, Figures 13a and 13b show an almost completely different distribution of dependencies. This is apparent when looking at any area of the two DSM solutions. The DSM sequenced by McCulley and Bloebaum contains 29 feedback marks and 67 crossovers[2] whereas that sequenced by the DaHA contains only 27 feedback marks and 41 crossovers.

In addition to the aforementioned comparisons, which all involved binary DSMs, the DaHA was applied to the numerical DSM studied by Qian et al. (Qian *et al.* 2011) which optimised Kusiak's objective function (Kusiak and Wang 1993) using meta-heuristics. The DaHA was found to give a different sequence to that obtained by Qian et al. but

[2]As defined by McCulley and Bloebaum (McCulley and Bloebaum 1996).

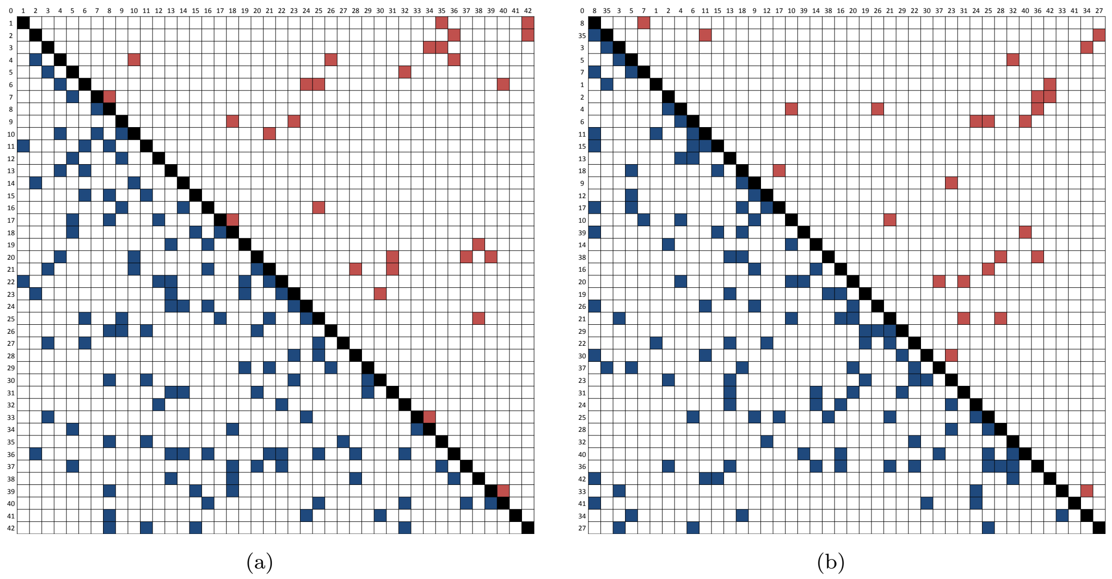


Figure 13.: (a) McCulley and Bloebaums's solution (McCulley and Bloebaum 1996) and (b) DaHA's solution.

with the same fitness according to Kusiak's objective function (Kusiak and Wang 1993). Figures 14a and 14b present the DSM solutions obtained by Qian et al. and the DaHA respectively. Figures 14a and 14b differ in their ordering of activities 5-23, which is apparent in the different distributions of dependencies between the two DSM solutions. However, both DSM solutions have an evaluation of 5.0 according to Kusiak's objective function (Kusiak and Wang 1993) where the value of all cells above the diagonal is summed.

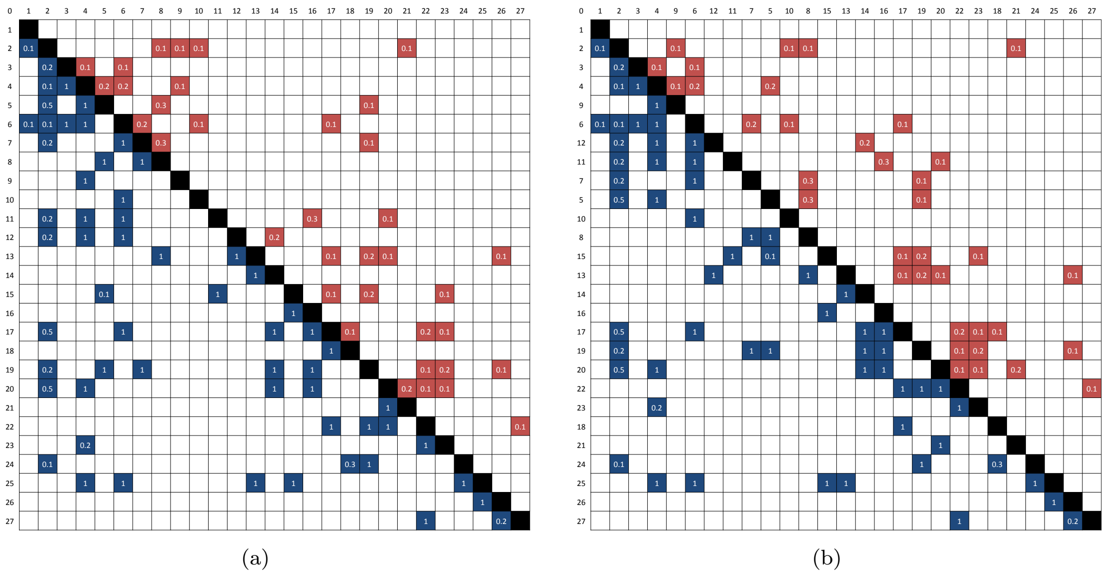


Figure 14.: (a) Qian et al.'s solution (Qian *et al.* 2011) and (b) DaHA's solution.

In summary, the DaHA has always achieved superior, or equal in the case of Scott and Qian et al., solutions in terms of fitness than other algorithms, thus demonstrating its efficacy as an enhanced GA sequencing the time-based DSM. Whilst the improvement on Todd's solution was minor and the DaHA was unable to improve on either Scott's or Qian et al.'s solutions, its ability to incorporate multiple objective functions and sequence DSMs of varying size and coupling density demonstrate the algorithm's versatility. It is noteworthy that the DaHA is the first GA that uses a hybridisation technique and that further refinement of the algorithm may lead to greater improvements on solutions found by other meta-heuristic techniques.

7. Conclusions

In this paper, an enhanced Genetic Algorithm, referred to as the DaHA, was presented as a new approach to sequencing the time-based DSM. In terms of contribution, the DaHA includes a new form of niching, which involves a population being divided into a number of sub-populations and crossover being restricted to individuals within the same sub-population. Subsequently, in an iterative manner, the current sub-populations are then hybridised to create the next set of sub-populations, with the aim of these producing better offspring and being able to locate their own local optimum. A further contribution is made through the development of a new ordinal-based selection method, referred to as the multiplier-driven selection method. This method provides each solution in a sub-population with a rank-based selection probability to encourage better individuals to be selected as parents thus increasing the likelihood of better offspring. Also, a lower limit for selection probability is employed to allow even the weakest solutions to be selected as parents with the aim of encouraging diversity and enabling a more thorough exploration of the solution space.

An investigation of the effect of various GA-based parameters on the performance of the DaHA has been summarised. Findings of this investigation showed that DSMs with lower coupling densities are more complex problems for GAs to solve and thus require larger population sizes. Furthermore, the investigation revealed that lower selection pressure than previously used in the literature can lead to increased solution fitness with non-prohibitive run times. More specifically, the combined effect of lowering the selection multiplier M and introducing a lower limit L was found to improve fitnesses by up to 5% with run times of under 60 seconds.

In terms of application, the DaHA has been shown to be a flexible and versatile algorithm, effective in solving a variety of different DSM problems. In a comparison with several other algorithms from the literature, the DaHA was able to achieve superior, or at least equal, results when sequencing the DSM. Improvements of up to 14% and 72% in fitness, according to the objective functions of the respective authors, were made on DSM problems previously sequenced with and without meta-heuristic optimisation respectively.

Finally, future work could be directed in three areas: (1) improving the performance of the DaHA, (2) widening the application of the DaHA, and (3) implementing and using the DaHA in practice. For example, in terms of (1), an investigation could be carried out into how the size of each sub-population, between generations, could be controlled dynamically such that those showing more promise in improving on the current solution could be increased in size, with other less promising sub-populations being reduced proportionately. Also, further analysis could be conducted to quantify how the various parameters in the DaHA should be tuned when sequencing DSMs of different sizes and coupling densities. In relation to (2), the DaHA could be applied to more numerical DSM problems than the single case presented briefly in this paper in order to establish if similar solution improvements can be achieved as obtained with the binary DSM problems considered. With regard to (3), research could be undertaken in how the DaHA could be implemented and used in practice by project managers to better sequence DSMs representing their projects. Such research would aim to demonstrate how complex, real-world projects could be better managed leading to appreciable improvements.

References

Ahmadi, R., Roemer, T.A., and Wang, R.H., 2001. Structuring product development processes. *European Journal of Operational Research*, 130 (3), 539–558.

Altus, S.S., Kroo, I.M., and Gage, P.J., 1996. A genetic algorithm for scheduling and decomposition of multidisciplinary design problems. *Journal of mechanical design*, 118 (4), 486–489.

Baldwin, C.Y. and Clark, K.B., 2000. *Design rules: The power of modularity.* Vol. 1. Mit Press.

Borjesson, F. and Hölttä-Otto, K., 2012. Improved clustering algorithm for design structure matrix. *In*: *Proceedings of the ASME 2012 international design engineering technical conferences and computers and information in engineering conference IDETC/CIE*, 12–15.

Brindle, A., 1981. Genetic Algorithms for Function Optimization. Thesis (PhD). University of Alberta.

Brooks, S.P. and Morgan, B.J., 1995. Optimization using simulated annealing. *The Statistician*, 241–257.

Browning, T.R., 2001. Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *Engineering Management, IEEE Transactions on*, 48 (3), 292–306.

Browning, T.R., 2010. On the alignment of the purposes and views of process models in project management. *Journal of Operations Management*, 28 (4), 316–332.

Browning, T.R. and Eppinger, S.D., 2002. Modeling impacts of process architecture on cost and schedule risk in product development. *Engineering Management, IEEE Transactions on*, 49 (4), 428–442.

Chen, L., Ding, Z., and Li, S., 2005. A formal two-phase method for decomposition of complex design problems. *Journal of Mechanical Design*, 127 (2), 184–195.

Cooper, K.G., 1993. The rework cycle: why projects are mismanaged. *PM network*, 7 (2), 5–7.

Davis, L., 1985. Job shop scheduling with genetic algorithms. *In*: *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Vol. 140.

De Jong, K.A., 1975. An Analysis of the Behavior of a Class of Genetic Adaptive Systems.. Thesis (PhD). Ann Arbor, MI, USA AAI7609381.

Dorigo, M. and Birattari, M., 2010. Ant colony optimization. *Encyclopedia of Machine Learning.* Springer, 36–39.

Eppinger, S.D., Bonelli, S., and Gonzalez, A.M., 2013. Managing Iterations in the Modular Real Estate Development Process. *In*: *Reducing risk in innovation: Proceedings of the 15th International DSM Conference Melbourne, Australia, 28-30 August 2013*, p. 37.

Gebala, D.A., Eppinger, S.D., *et al.*, 1991. *Methods for analyzing design procedures.* Sloan School of Management, Massachusetts Institute of Technology.

Glover, F., 1989. Tabu search-part I. *ORSA Journal on computing*, 1 (3), 190–206.

Glover, F., 1990. Tabu searchpart II. *ORSA Journal on computing*, 2 (1), 4–32.

Goldberg, D.E. and Holland, J.H., 1988. Genetic algorithms and machine learning. *Machine learning*, 3 (2), 95–99.

Grefenstette, J.J., 1986. Optimization of control parameters for genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 16 (1), 122–128.

Helmer, R., Yassine, A., and Meier, C., 2010. Systematic module and interface definition using component design structure matrix. *Journal of Engineering Design*, 21 (6), 647–675.

Holland, J.H., 1975. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.*. U Michigan Press.

Jung, S. and Simpson, T.W., 2014. A Clustering Method Using New Modularity Indices and Genetic Algorithm with Extended Chromosomes. *In*: M.M.D.M.S. Franck Marle Marija Jankovic and U. Lindemann, eds. *Risk and change management in complex systems: Proceedings of the 16th International DSM Conference* Carl Hanser Verlag, Munich.

Karniel, A. and Reich, Y., 2009. From DSM-based planning to design process simulation: a review of process scheme logic verification issues. *Engineering Management, IEEE Transactions on*, 56 (4), 636–649.

Kusiak, A. and Wang, J., 1993. Decomposition of the design process. *Journal of Mechanical Design*, 115 (4), 687–695.

Laumanns, M., Zitzler, E., and Thiele, L., 2000. A unified model for multi-objective evolutionary algorithms with elitism. *In*: *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, Vol. 1, 46–53.

Liu, D., *et al.*, 2014. Short-term wind speed forecasting using wavelet transform and support vector machines optimized by genetic algorithm. *Renewable Energy*, 62, 592–597.

Mahfoud, S.W., 1995. Niching methods for genetic algorithms. *Urbana*, 51 (95001), 62–94.

McCulley, C. and Bloebaum, C., 1996. A genetic tool for optimal design sequencing in complex engineering systems. *Structural Optimization*, 12 (2-3), 186–201.

Meier, C., Yassine, A.A., and Browning, T.R., 2007. Design process sequencing with competent genetic algorithms. *Journal of Mechanical Design*, 129 (6), 566–585.

Mühlenbein, H., 1992. How Genetic Algorithms Really Work: Mutation and Hillclimbing.. *In*: *PPSN*, Vol. 92, 15–25.

Murata, T. and Ishibuchi, H., 1994. Performance evaluation of genetic algorithms for flowshop scheduling problems. *In*: *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, 812–817.

Otto, K., Hölttä-Otto, K., and Simpson, T.W., 2013. Linking 10 years of modular design research: alternative methods and tool chain sequences to support product platform design. *In*: *ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, V03BT03A035–V03BT03A035.

Petz, A., *et al.*, 2014. Modeling and Simulation of Service Systems with Design Structure and Domain Mapping Matrices. *In*: M.M.D.M.S. Franck Marle Marija Jankovic and U. Lindemann, eds. *Risk and change management in complex systems: Proceedings of the 16th International DSM Conference* Carl Hanser Verlag, Munich.

Qian, Y., *et al.*, 2011. A novel approach to DSM-based activity sequencing problem.

Engineering Management, IEEE Transactions on, 58 (4), 688–705.

Roberge, V., Tarbouchi, M., and Labonté, G., 2013. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. *Industrial Informatics, IEEE Transactions on*, 9 (1), 132–141.

Rogers, J., 1996. DEMAID/GAAn Enhanced Design Managers Aid for Intelligent Decomposition (Genetic Algorithms). *In*: *Proceedings of the Sixth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Sept*, 4–6.

Scott, J.A., 1999. A strategy for modelling the design-development phase of a product. .

Sörensen, K. and Glover, F., 2013. *Encyclopedia of Operations Research and Management Science*. Springer.

Steward, D.V., 1981a. The design structure system: a method for managing the design of complex systems. *IEEE transactions on Engineering Management*, (EM-28).

Steward, D.V., 1981b. *Systems analysis and management: structure, strategy and design*. PBI New York.

Syswerda, G., 1991. Schedule optimization using genetic algorithms. *Handbook of genetic algorithms*.

Tarjan, R., 1972. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1 (2), 146–160.

Todd, D., 1997. Multiple criteria genetic algorithms in engineering design and operation. Thesis (PhD). Citeseer.

Vidal, T., *et al.*, 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40 (1), 475–489.

Vose, G.E.L.M.D., 2014. Deceptiveness and Genetic Algorithm Dynamics'. *Foundations of Genetic Algorithms 1991 (FOGA 1)*, 1, 36.

Warfield, J.N., 1973. Binary matrices in system modeling. *Systems, Man and Cybernetics, IEEE Transactions on*, (5), 441–449.

Whitfield, R., Duffy, A., and Kortabarria, L., 2005. Identifying and evaluating parallel design activities using the design structure matrix. .

Whitfield, R.I., Smith, J.S., and Duffy, A.B., 2002. Identifying component modules. *In*: *Artificial Intelligence in Design02*, 571–592.

Whitfield, R.I., *et al.*, 2003. Efficient process optimization. *Concurrent Engineering*, 11 (2), 83–92.

Whitley, D., 2014. An executable model of a simple genetic algorithm. *Foundations of genetic algorithms*, 2 (1519), 45–62.

Yassine, A.A., *et al.*, 2000. Do-it-rightfirst-time (DRFT) approach to DSM restructuring. *In*: *Proc. ASME 2000 Int. Design Engineering Technical Conf.(DTM)*.

Yu, T.L., Yassine, A.A., and Goldberg, D.E., 2003. A genetic algorithm for developing modular product architectures. *In*: *ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 515–524.

Yu, T.L., Yassine, A.A., and Goldberg, D.E., 2007. An information theoretic method for developing modular architectures using genetic algorithms. *Research in Engineering Design*, 18 (2), 91–109.

Appendix A. Comparison of solutions

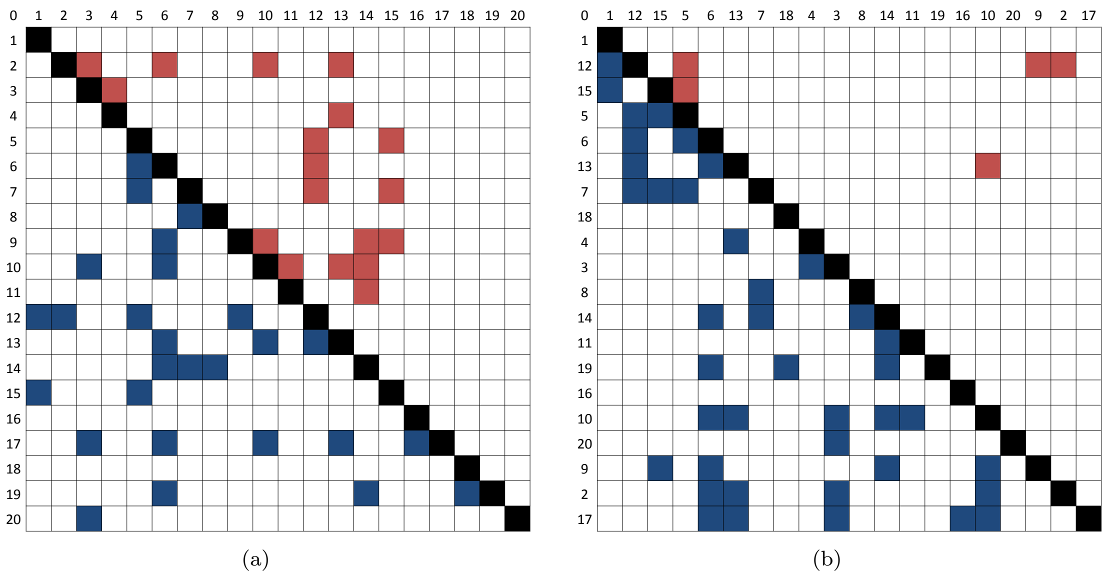


Figure A1.: (a) Steward's solution (Steward 1981b) and (b) DaHA's solution.

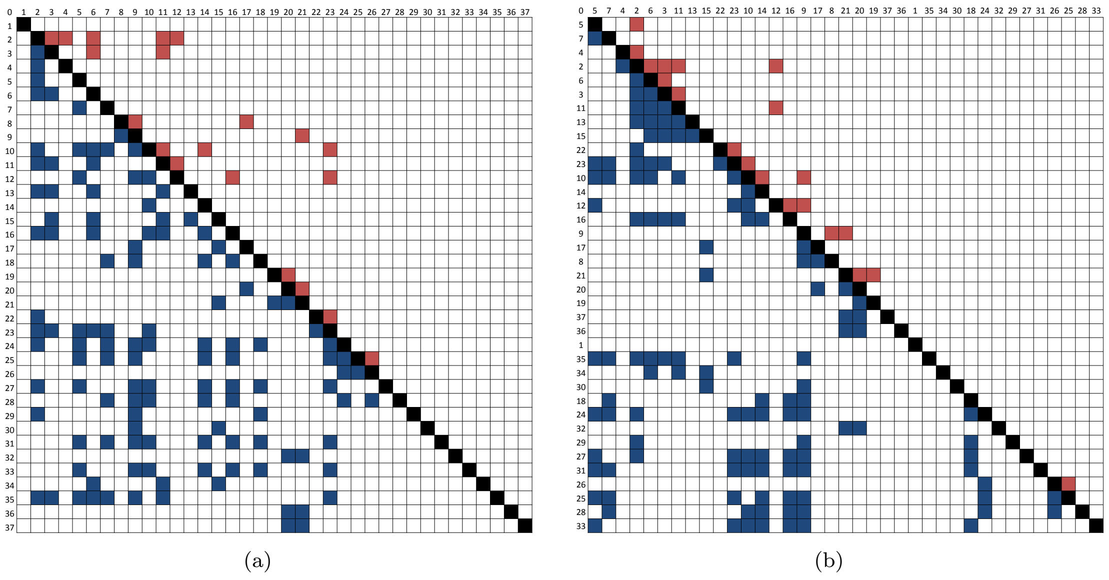


Figure A2.: (a) Gebala and Eppinger's solution (Gebala *et al.* 1991) and (b) DaHA's solution.