Single Machine Total Weighted Tardiness Problem

Application of Genetic Algorithm and Ant Colony Optimization Hybrid for SMTWTP

December 15, 2018

Luca Ostertag-Hill, Tom Lucy, Jake Rourke

Nature Inspired Computation
Final Project

Abstract—**This project examined the performance of a Genetic Algorithm (GA) and Elitist Ant System (EAS) nature inspired hybrid for the single machine total weighted tardiness problem (SMTWTP). The objective of SMTWTP is, given a set of jobs, each of which has a processing time, a weight, and must be completed before a given due date, order the jobs in a way to minimize the total weighted tardiness. EAS is one of many nature-inspired techniques created to solve search optimization problems and has several features that make it useful in solving SMTWTP. Based on the foraging behavior of ants, the EAS algorithm iteratively builds solutions based on a stochastic selection process. The variation of EAS implemented in this project is the EAS, which deposits an increased amount of pheromone on the best tour found so far to stimulate increased exploitation. GA uses an iterative process of selection, crossover, and mutation to converge on a solution in the search space. GA is used in series with EAS to perform local search after EAS has found several good solutions in the global scope. Three approaches, EAS by itself, GA by itself, and the hybrid of the two, were evaluated for their effectiveness on three different problem instances. GA had the best performance on all three problems, the hybrid approach had similar performances, and EAS had significantly worse performances.**

I. Introduction

IN this project, we seek to find optimal solutions to the Single Machine Total Weighted Tardiness Problem (SMTWTP) by using a hybrid nature-inspired computing method that combines Elitist Ant System with a Genetic Algorithm. The inspiration for this hybridization comes from Kyriklidis et al. 2014 in which a similar hybrid algorithm was used to solve the resource-leveling problem, which is a different, but similar, problem.

The SMTWTP comes from project management and factory organization. In each problem instance, there is a set of jobs that need to be completed and a single machine that can complete the jobs. Each job has a processing time, a weight, and a due date. A solution to the problem instance requires that all jobs must be completed. A solution can be evaluated for its optimality based on the total weighted tardiness of all jobs, which is the tardiness of a given job times its weight. An optimal solution represents the tardiness-minimizing ordering of jobs for processing on the single machine.

The hybrid algorithm used to solve instances of the SMTWTP will be a combination of Elitist Ant System (EAS) and a Genetic Algorithm (GA). EAS will seek to perform a global search of the solution space, i.e. it will be run multiple times on a problem instance and hopefully develop multiple good solutions. After the good solutions have been found by EAS, the GA will take the set of good solutions as a starting population. The algorithm will run separately on each solutions produced by EAS, and therefore the GA will be performing a local search of each good solutions region in the solution space. The approach of this hybridization comes from the determination that the traditional ACO algorithm, while efficient at finding a good solution, does poorly on local search. This is evidenced by the numerous variations on the traditional Ant System that attempt to exploit local search more, such as Ant Colony System. However, this hybridization eliminates the largest weakness of ACO by making it wholly a global search algorithm. Then GA, which is well suited through mutation to make small changes in the search space, performs local search.

The objective of this project was to determine which algorithm, ACO, GA, or the hybrid, performed better at solving SMTWTP. In order to compare these algorithms we conducted preliminary testing on the parameter settings of each algorithm. We performed tests on problems of varying size, including files with 40, 50, and 100 jobs. These tests are given in Table II and the medians in Table III. In comparing the algorithms on small problems (40 and 50 jobs), GA and the hybrid easily outperformed ACO. Though the difference was not large, pure GA did perform better than the hybrid. On the larger problem file, GA pulled father away from the hybrid, while ACO clearly showed itself as the worst of the three algorithms.

The following is a road-map for the rest of the paper. In section two, there is a more detailed description of SMTWTP. In section three, elitist ant system is explained in detail. In section four, genetic algorithms are explained in detail. In section five, we lay out our experimental methodology. In section six, there is a full review and analysis of our results. In section seven, possible avenues of further work are discussed. Finally, in section eight, we provide a summary of our conclusions.

II. Single Machine Total Weighted Tardiness Problem

The Single Machine Total Weighted Tardiness Problem (SMTWTP) is a canonical problem in project management and factory organization. In an instance of a problem, there are several components. First, as implied by the name, there is a single machine available for use. This machine is a general machine, and is the only thing that can complete a job. Next, there is a set of n jobs, each of which has a due date and a weight, and need to be completed. The total time taken to complete all jobs in the set is fixed, therefore we seek to minimize the total weighted tardiness of the jobs rather than the time to taken for total completion.

Given this problem definition, a solution to an instance of SMTWTP is represented as a vector of jobs. All jobs in the problem must be included in the solution vector. As we seek to minimize the total weighted tardiness of the jobs, the optimality of the solution vector is based on its total weighted tardiness, the cross product of the each jobs tardiness and its weight.

The tardiness of a schedule is defined as the time past the due date taken to complete a job. Therefore, if a job i is started at time T, has a processing time of t_i, and a due-date of d_i the tardiness of that job is defined as

$$tardiness = \max(0, (T + t_i) - d_i)$$

If job i finishes before its due date, there is no consideration for its earliness, but it receives a tardiness of 0. Job is tardiness must also be weighted by the jobs weight, w_i, as some jobs are more important than others. As such, the total tardiness of all jobs is defined as

$$totalTardiness = \sum_{i=1} w_i tardiness_i$$

III. Elitist Ant System

Elitist Ant System (EAS) works similarly to Ant Colony Optimization (ACO), but with a few slight changes. As such, ACO will first be explained and then EAS will be explained as it deviates from ACO. As the name suggests, ACO seeks to optimize a solution through the search space in much the same way that ants find an optimal solution path. In finding an optimal path, ants explore the area around their nest for food, carry food back to the nest, and lay pheromones on this path, these pheromones then guide other ants to the found food. ACO replicates this behavior by leaving pheromone behind to guide the ants more strongly to better solutions. The pheromone levels are stored in a pheromone matrix, that is later utilized in the selection process. In terms of the SMTWTP, the pheromone matrix is a set of vectors, each of which represents a job. An individual vector for a job contains the pheromone levels for all possible paths from that job. Thus the matrix is a 2d job by job array, where any element in the matrix represents the pheromone level of the path between the first job to the second job. Further traditional ACO utilizes a heuristic in combination with the pheromone matrix.

$$eta_i = 1/dueDate_i$$

First in ACO, a hive of ants is generated with iteratively created solutions. These solutions are then evaluated for their fitness. The fitness of each solution is the weighted tardiness. When searching for a minimum cost solution, solutions of lower values will have higher fitness. After the fitness evaluation of a solution, pheromones are then deposited in the vectors of pheromone matrix corresponding to path between two consecutive jobs relative to its fitness - i.e. the quantity of pheromone indicates how good the solution is. The amount of pheromone to be deposited at a starting time of ant ks solution is given by:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{L_k} & \text{if } (i,\, j) \text{ in tour} \\ 0 & \text{otherwise} \end{cases}$$

Where $\Delta\tau_{ij}^{k}$ is equal to the reciprocal of the cost of ant ks solution if the path from job i to job j is included in ant ks tour, otherwise it is 0. After pheromone has been applied to all solutions, paths that more frequently appear in better solutions will have higher pheromone levels.

After this initial pheromone laying, ants then construct new solutions, with ants choosing a path based on attraction to the present pheromones. To create a path, an ant uses stochastic solution construction of all possible operations at the present time. Therefore, ant k chooses the next operation i with start time j as the next stop with probability:

$$\rho_{ij}^{k} = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum\limits_{k \in N_k} \tau_{ik}^{\alpha}\eta_{ik}^{\beta}}$$

In this formula, τ_{ij} is the pheromone concentration on the workflow path from job i to job j, η_{ij} is the heuristic of job j, N_k is the set of jobs not yet visited by ant k, and α and β are positive real constants that scale the effects.

After the ants have completed their second solutions, the processes of evaluation and pheromone application begin once more, but a third process also begins. Every iteration, old pheromone begins to evaporate off of the matrix. This is done to allow past mistakes in solution search to be overcome, and increase reliance on more recently found results. The evaporation is governed by the evaporation factor ρ, which is some number between zero and one. All of this combines to yield an overall iteration pheromone change of:

$$\tau_{ij} = \tau_{ij}(1-\rho) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

This process will continue until a certain number of iterations has been reached.

Given this explanation of ACO, it will now be explained how EAS differs. First, it is important to remember that EAS is similar in most ways, both in theory and in implementation. However, the most important differences come in the pheromone application. EAS favors the best solution so far over all other tours and applies pheromones accordingly. The best solution so far is defined as the best solution that has been found over all iterations. To implement this favoritism, EAS puts more pheromone on only the best solutions so far. This leads to an overall iteration pheromone change of:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \tau_{ij}^{k} + e\Delta\tau_{ij}^{bsf}$$

Where τ_{ij}^{bsf} is the reciprocal of the length of the tour leg from node i to node j in the best tour so far, otherwise it is 0.

IV. Genetic Algorithms

The conceptual foundation of Genetic Algorithm comes from the intertwined process of natural-selection and evolution. The algorithm is used to solve optimization and search problems through an iterative process that evolves a population of individuals. Each iteration of the algorithm is called a generation and uses three nature-inspired techniques to evolve the population towards an optimal solution. These three techniques are selection, recombination (often referred to as crossover), and mutation. The use of all three of these techniques allows the Genetic Algorithm to strike a balance between exploration of the search space and exploitation of the best found solutions so far. This balance between exploration and exploitation is critical, as it allows Genetic Algorithms to overcome local maximums yet handle a large search space. A basic outline of the Genetic Algorithm is outlined below and the three evolutionary techniques are expanded on later.

1) Generate a starting population of n individuals, wherein each element of each individual is randomly generated with equal probability.
2) Evaluate and score each individual based on the fitness function.
3) Select n individuals without duplicates for the breeding pool, favoring individuals with a higher fitness.
4) Perform crossover on parent pairs with some defined probability. Otherwise move on unchanged. This produces a new population of size n.
5) Perform mutation on each element of each individual in the new population with some defined probability.
6) Iterate over steps 2 through 6 until an optimal solution is found or the defined number of generations allowed is over.

A. Selection

Selection is the process of choosing some number of individuals from the current population to act as a breeding pool for the next generation. The method for selecting these individuals is fitness-based. This means that individuals with a higher fitness are more likely to be chosen, giving them a greater opportunity to exert their influence on the next generation. The fitness of an individual is representative of how close the individual is to the optimum solution of the problem. When working with SMTWTP, the fitness of each individual is the total cost of the solution (total weighted tardiness). The bias towards fitter individuals is representative of exploitation and is necessary to evolve the population towards an optimal solution. Duplicate individuals are not allowed in the breeding pool, in order to force exploration of the search space. Fitness-based selection is also probabilistic, so it inherently provides some randomness. In this project we will be using tournament selection to determine the breeding pool.

Tournament selection works by randomly choosing M individuals from the population and taking the best k of those. The k individuals chosen are the k most fit individuals in the random set of M. Typical values for M and k are 2 and 1. This means that 2 random individuals are chosen from the population and the fitter individual is placed in the breeding pool. This occurs until the breeding pool is full.

B. Recombination

Crossover uses the breeding pool of individuals found through selection to create the next generation of individuals. Individuals from the breeding pool are randomly paired up in groups of 2 (representative of parents) and are combined to produce offspring. Crossover occurs probabilistically between each pair of individuals to introduce randomness. Typical values for crossover range from $[0.6, 0.9]$. If crossover does not occur, the two offspring are clones of the parents (parent solutions move onto next generation). This allows for big steps to be made in the solution space and allows Genetic Algorithms to escape local maximums. In this project we will be using order-one crossover. In order-one crossover, a random cut point is selected for each parent pair and indicates how many jobs child 1 receives from parent 1 and how many jobs child 2 receives from parent 2. The example below uses a cut point of 2.

$$Parent_1 = 4, 3, 5, 2, 1 \quad Parent_1 = 4, 3|5, 2, 1$$

$$Parent_2 = 2, 1, 4, 3, 5 \quad Parent_2 = 2, 1|4, 3, 5$$

All of the jobs to the left of the cut point are passed down to the corresponding child.

$$Child_1 = 4, 3, x, x, x$$

$$Child_2 = 2, 1, x, x, x$$

The rest of the values are filled in order from the opposite parent.

$$Child_1 = 4, 3, 2, 1, 5$$

$$Child_2 = 2, 1, 4, 3, 5$$

Keeping the values in order maintains the shape of the solution, by generally keeping jobs in a similar position as before, with slight variance introduced for exploration.

C. Mutation

Mutation is used as a technique to make tiny changes to individuals in the population. Mutation occurs on each element of each child with some small probability. Typical values for mutation range from $[0.001, 0.02]$, but for SMTWTP larger values are generally used to increase exploration. Mutation is probabilistic, but occurs on a very small scale. This allows Genetic Algorithms to accomplish little steps in the solution space by conducting local search around itself. For SMTWTP there are two distinct techniques for mutation and both are implemented in our project. The first type of mutation is standard in GA. Two indices in the childs workflow are randomly selected and swapped. This allows for broader mutation in the workflow, as jobs that start early can be swapped with jobs on the other end of the array. The second type of mutation is a range reversal mutation. Again, a random index is selected and a random range size between two and four is chosen. Starting at the index, the next two to four elements are reversed. This allows for more local mutation, as only values that are near each other can be altered.

V. Experimental Methodology

The goal of this project is not necessarily to optimize any one algorithm or method to solve the Single Machine Total Weighted Tardiness Problem (SMTWTP), but rather to compare different approaches and determine which approach was the most effective. The three considered are: EAS alone, GA alone, and our combined EAS-GA hybrid algorithm. In the EAS alone approach, EAS attempts to solve the problem instance by itself, performing both global and local search. In the GA alone approach, Ga also attempts to solve the problem instance by itself, performing both global and local search. However, in the hybrid approach, EAS performs a global search of the solution space and then passes a set of good solutions onto GA, which performs the local search. To fairly compare the three approaches, it is important to standardize the number of iterations that each approach gets to work with. Our baseline number of iterations in the project is 300. When EAS or GA operate alone, they receive the full 300 iterations. However, when the hybrid operates, it receives only 150 iterations for EAS and 150 iterations for GA. It is also important to note how EAS is producing its solution. The produced best solution does not represent the best solution found by a single run of EAS, but rather the best solution out of 100 runs of EAS. This is true for both the EAS alone approach and the hybrid approach. On the other hand, GAs produced solution is the product of only a single run.

Beyond the fixed number of iterations allotted to each approach, other necessary parameters were set constant. All of these constant values were found through a light set of preliminary testing. For EAS, the number of ants was fixed to 50, the alpha value was fixed to 1, the beta value was fixed to 6, the rho value was fixed to 0.001, and the elitism factor was fixed to 100. For GA, the population size was fixed to 100, the mutation probability was fixed to 0.6, and the crossover probability was fixed to 0.9. The number to which the population size was fixed is significant in that it determines the number of times EAS runs before selecting a best. As the population with which GA starts is composed entirely of EASs best solutions, EAS needs to repeat itself until it has filled GAs population. These fixed parameters were held constant across all runs of their respective algorithms. This includes both when EAS or GA were used on their own as well as when they were used as part of the hybrid approach.

To best compare the three approaches, we selected a sweet of test problems on which to run them. These test problems included one 40-job problem, one 50-job problem, and one 100-job problem. Each was specifically selected from the set of test problems available to us for both its generality and its intricacies. For example, the 100-job problem presents some nuances in that many jobs have a due date of 0. This means that no matter when the job is completed, it will be tardy. Therefore, the importance of completing it early is much more dependant on its weight than any other factor. The three test files used, as well as their optimal solutions, are listed in Table 1.

File Name	*Optimal Cost*
40.1.txt	913
50.1.txt	2134
100.1.txt	898925

TABLE I
OPTIMAL SOLUTION COSTS FOR THREE DIFFERENT SIZED FILES (NUMBER OF JOBS: 40, 50, 100).

Each approach was run on each of the test problems ten times. This allows us to analyze the results in multiple ways. First, we can take the median score of each approach on a particular test problem and directly compare its efficacy on the problem. Second, we can consider the full distribution of performances on a test problem and see if certain approaches skewed toward good results or bad results, if there were any outliers of significance, or if there were scores around which the approaches got stuck. Third, by running the approaches a significant amount of times on a range of test problems, we are able to make a full determination about which approach is better on the SMTWTP in general. Possibly, our results will reveal a three-way tie of the algorithms, i.e. each one gave the best performance on one of the three problems. However, if this is not the case, one approach will have done better on the majority of problems and, by looking at the significance of the margins, we can determine whether or not it is likely to be the best of the three approaches.

VI. RESULTS

Ten trials were performed of each algorithms on each test file. These trials are presented in Table II below. The lowest cost solution found in each trial is bolded. Each of the lowest cost solutions for each file came from the GA algorithm. The lowest cost solution found by pure GA and the hybrid were very consistent across the ten trials. The spread of solutions for GA on the 40 job problem was only 930 to 1018 and for the 50 job problem 2134 to 2345 for the 50 job problem. The spread of solutions for the hybrid on the 40 job problem was only 956 to 1056 and for the 50 job problem 2184 to 2395 for the 50 job problem. The pure EAS algorithm showed more inconsistency with a spread of 1236 to 1629 for the 40 job problem and 2341 to 2822 on the 50 job problem.

Num Jobs	*Algorithm*	*Trial 1*	*Trial 2*	*Trial 3*	*Trial 4*	*Trial 5*	*Trial 6*	*Trial 7*	*Trial 8*	*Trial 9*	*Trial 10*
40	EAS	1228	1531	1591	1629	1441	1555	**1236**	1570	1261	1515
	GA	956	956	980	1012	956	1012	956	956	1018	**930**
	Hybrid	**956**	956	1012	980	1012	956	1032	956	1056	1015
50	EAS	2573	2445	**2341**	2345	2537	2682	2822	2437	2583	2450
	GA	**2134**	2345	2247	2184	2192	2345	2345	2134	2345	2345
	Hybrid	2395	2345	2345	**2184**	2247	2184	2314	2295	2387	2184
100	EAS	1311979	1315845	1331449	1326128	1328269	1326368	1310767	1325067	1316147	**1307325**
	GA	926519	930135	927623	927417	926179	923565	921760	**921739**	927855	921800
	Hybrid	946812	929797	935487	932003	939247	940529	934377	945175	944754	**928284**

TABLE II
SIMULATION DATA FOR THREE ALGORITHMS (EAS, GA, HYBRID) ON THREE TEST FILES.

Num Jobs	*Algorithm*	*Median*	*Optimal*
40	EAS	1523	913
	GA	956	
	Hybrid	996	
50	EAS	2493.5	2134
	GA	2296	
	Hybrid	2304.5	
100	EAS	1320607	898925
	GA	926349	
	Hybrid	937367	

TABLE III
MEDIAN VALUES FOR SIMULATION DATA FOR THREE ALGORITHMS (EAS, GA, HYBRID) ON THREE TEST FILES.

The median values for these simulations are presented in Table III above with the optimal cost found by Crauwels, Potts, and Van Wassenhove (1998) and Congram, Potts, and van de Velde (1998). Both the GA and hybrid came relatively close to the optimal on all three problem files. The GA median was within 5%, 8%, and 3% for the 40, 50, and 100 job problems respectively. The hybrid median was within 9%, 8%, and 4%. Alternatively, the pure EAS only came within 67%, 17%, and 47%. This shows that EAS was easily the worst algorithm for SMTWTP and GA barely edged the hybrid for best overall performance. The best solutions that were found overall (by GA) were 1.8%, 0.0%, and 2.5% worse than the known optimal costs. This means that for the 50 job problem, our GA found a workflow with equivalent cost to the optimal known solution.

The distribution for the simulations on the 40 job problem are displayed on the next page in Figure 1. The figure clearly shows that EAS found the worst solutions out of the three and that its spread was significantly larger than either GA or the hybrid. The GA and hybrid distribution mirror each other, but the GA distribution is shifted to the left, showcasing its performance over the hybrid. Both the GA and hybrid have a normal distribution for the 40 job problem.

The distribution for the simulations on the 50 job problem are displayed in Figure 2. The three distributions are much closer than they were in the 40 job problem. EAS remains spread out across the board, though its best values are able to outperform the middle class of values in both GA and the hybrid. The distributions for GA and the hybrid no longer represent a bell curve. Instead the distribution for the hybrid has multiple maximums and the GA distribution represents a positive sloping distribution. This is likely due to the fact that around a solution cost of 2300, it was difficult to improve. However, once improvement was made, it was easier to make further improvements up towards the optimal solution.

The distribution for the simulations on the 100 job problem are displayed in Figure 3. The figure clearly shows that pure GA performs the best with SMTWTP. All but one of the trials of GA beat out the best found solution for the hybrid. Further, the solutions found by GA showed little variance across multiple trials, suggesting that its mutation allows it to constantly overcome hurdles in the solution space. Preliminary testing with GA showed that the use of just one of the mutation methods gave results

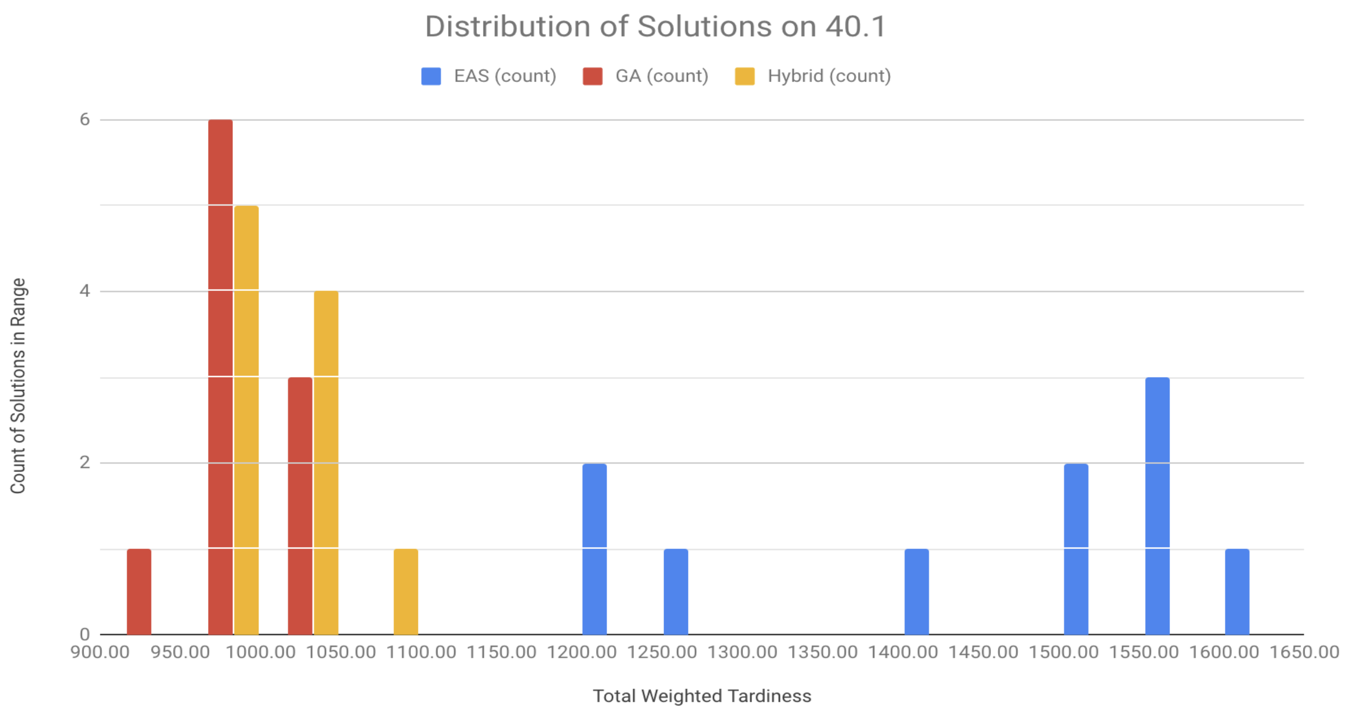

Fig. 1. Simulation of algorithms on problem with 40 jobs.

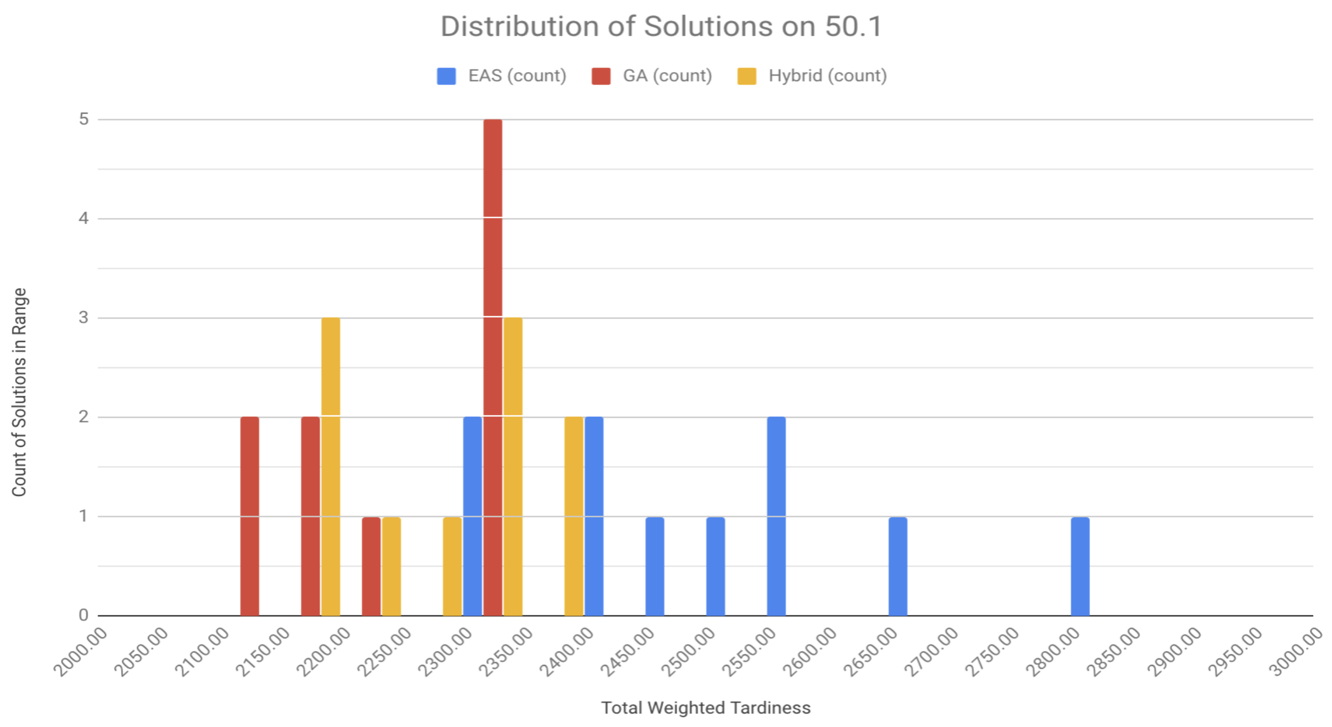

Fig. 2. Simulation of algorithms on problem with 50 jobs.

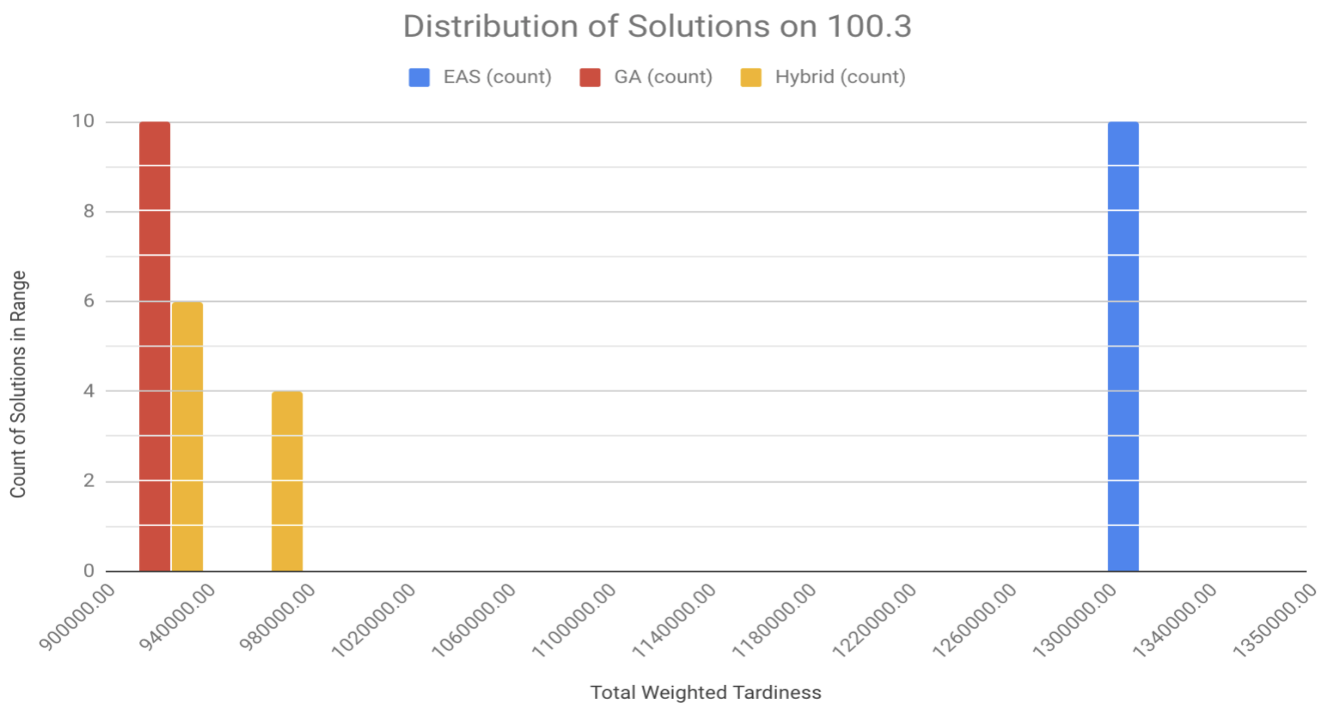

Fig. 3. Simulation of algorithms on problem with 100 jobs.

worse than the hybrid. The combination of the two mutation techniques allowed the GA to excel and constantly improve.

In the end, the pure GA algorithm performed better than the hybrid of EAS as a global search and GA as a local search. The pure EAS algorithm fell far behind in performance in comparison to either the GA or the hybrid. The EAS algorithm performed poorly due to its inability to perform good local search. Reviews of the best found solution throughout iterations in EAS shows that the algorithm easily gets stuck at a solution and cannot escape. On the other hand, GA possesses a purely local search technique in mutation. While mutation probability is generally set between $[0.001, 0.02]$ we used a value of 0.6. Further, because two techniques were employed, there was a 0.84% probability that at least one form of mutation was performed on a child. Local search is evidently critical to attaining good solutions for SMTWTP. Although the hybrid allows for decent solutions to be passed to the GA algorithm, the GA algorithm is run for fewer generations (to keep cost of execution the same across the algorithms). This minimizes how much local search the hybrid can perform in a problem where local search is key.

VII. Further Work

There are three significant ways in which we could expand upon our project and further our understandings. First, we could change the order of the hybrid algorithm. Second, we could implement a completely different version of Ant Colony Optimization (ACO) in our hybrid algorithm. Third, we could employ more rigorous testing to find the optimal parameters of EAS and GA before the comparative experimentation. It is likely that any of these options would reveal new findings and possibly offer additional insight to our current findings.

In changing the order of the hybrid algorithm, we would just be running the GA first, rather than our current order. By doing this, we would be using GA to perform the global search and EAS to take the produced decent solutions and perform a local search. This may be very ineffective, as GA produces very good solutions from a random starting point. However, it could also prove to be great as EAS might provide the final push to get GAs good solutions to optimal.

To accomplish the goal of exploring different versions of ACO, we would most likely implement Ant Colony System (ACS) for our hybrid. The ACO component of our project only used EAS. It is entirely possible that certain aspects of ACS, such as the probabilistic creation of greedy tours, could be very applicable to the SMTWTP. By exploring the performance of a different type we may see improvement in our hybrid or gain new knowledge about it.

To optimize our parameters better, we could conduct extensive, dedicated tests for each parameter and each algorithm. Our admittedly light optimizations of our parameters suggest that there is more work to be done in this area. EAS performed much worse than GA, and it is entirely possible that a better set of initial parameters would improve its performance. Improving the performance of EAS would not only benefit our analysis of EAS on its own, but may benefit the hybrid approach enough for it to consistently perform better than GA.

VIII. Conclusions

This project was conducted to test the efficacy of a Genetic Algorithm (GA) and Elitist Ant System (EAS) hybrid on the Single Machine Total Weighted Tardiness Problem (SMTWTP). Each individual algorithm was tested against the hybrid on files of size 40, 50, and 100 jobs. The data from these tests shows that pure GA outperformed the hybrid algorithm in finding a lower cost solution, while EAS failed to find even decent solutions for two of the three problem files. The performance of GA can be attributed to the high mutation probability and two mutation techniques utilized that allowed for extensive local search in the solution space. Conversely, the lack of a local search in EAS produced poor results. Lastly, during our testing of the GA on the problem with 50 jobs, our algorithm actually found a workflow with a weighted tardiness cost that matched that of the best known solution. This highlights how well the GA performed and shows that the hybrid did generally find good solutions. This suggest that the hybrid may in fact be worth utilizing on other optimization problems that dont have such a heavy focus on local search.

IX. References

1) Stephen Majercik In-Class Notes
2) *Hybrid nature-inspired intelligence for the resource leveling problem.* Christos Kyriklidis, Vassilios Vassiliadis, Konstantinos Kirytopoulos, Georgios Dounias. Volume 14, Issue 3. 13 May 2014. https://link.springer.com/article/10.1007/s12351-014-0145-x.
3) *Weighted tardiness problem files.* OR-Library. J E Beasley. http://people.brunel.ac.uk/ mastjjb/jeb/orlib/wtinfo.html.
4) *A genetic algorithm for the Flexible Job-shop Scheduling Problem.* F.Pezzellaa, G.Morgantia, G.Ciaschettib. Volume 35, Issue 10, October 2008.
5) *Solving the flexible job shop problem by hybrid metaheuristics-based multiagent model.* Houssem Eddine Nouri, Olfa Belkahla Driss, Khaled Ghdira.Volume 14, Issue 1, 16 May 2017.
6) *Flexible Job Shop Scheduling Problem Using an Improved Ant Colony Optimization.* Lei Wang, Jingcao Cai, Ming Li, Zhihu Liu. Volume 2017, Article ID 9016303. 26 January 2017.