Genetic Algorithms for the Sailor Assignment Problem

Deon Garrett, Joseph Vannucci, Rodrigo Silva, Dipankar Dasgupta
Department of Computer Science
University of Memphis
Memphis, TN 38152
{jdgarrtt,jvannucc,rsilva,dasgupta}@memphis.edu

James Simien
Research Analyst
Naval Personnel Research, Studies and Technology
Millington, TN 38055
james.simien@navy.mil

ABSTRACT

This paper examines a real-world application of genetic algorithms – solving the United States Navy's Sailor Assignment Problem (SAP). The SAP is a complex assignment problem in which each of n sailors must be assigned one job drawn from a set of m jobs. The goal is to find a set of these assignments such that the overall desirability of the match is maximized while the cost of the match is minimized. We compare genetic algorithms to an existing algorithm, the Gale-Shapley algorithm, for generating these assignments and present empirical results showing that the GA is able to produce good solutions with significant savings in cost. Finally, we examine the possibility of using the GA to generate multiple different solutions for presentation to a human decision maker called a detailer, and we show that the GA can be used to provide a sample of good solutions.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Global Optimization

General Terms

Algorithms

Keywords

Genetic Algorithms, Assignment Problem, CHC

1. DESCRIPTION OF THE PROBLEM

Many real-world problems can be modeled as assignment problems. Given two sets, the assignment problem is to match each element of set A with a single element of set B such that some criteria of desirability is maximized. The simplest example of such a problem is the so-called *stable marriage problem*. In this problem, we are given a set of n men and a set of n women. The problem is to find the optimal stable set of marriages between the men and the women with respect to the stated preferences of the individuals. A stable set of marriages is defined to be a set of marriages in which the following can not occur: A is married to B and C is married to D while A prefers D to B and D prefers A to C. If this situation did occur, then obviously, A and D would leave their partners for each other. An optimal stable marriage is a stable marriage in which each person is at least as happy as they would be in any other stable set of marriages.

Another example of such a problem is the United States Navy's sailor assignment problem [9, 11]. Every two years, each sailor in the Navy is required to change jobs. As a result, at any given time, there are many sailors who require assignment to a new job. The problem the Navy faces is to find a set of assignments, also called a match, of sailors to jobs which keeps the sailors happy and maintains fleet readiness while minimizing the cost of implementing those assignments. Many factors go into determining a good set of assignments. The Navy must ensure that not only are the sailors and commanders satisfied with the assignment, but also that the match can be implemented within a fixed budget. If the resulting cost is too expensive, then the Navy cannot adequately maintain its other priorities. However, if the assignment is overly focused on costs, it may be that the sailors are unhappy with their assigned jobs. This could lead to a decrease in morale, followed by decreased retention rates and other problems.

Formally, the problem may be defined as follows. Maximize

$$\sum_{i=1}^{N} \sum_{j=1}^{M} \mathcal{F}_{i,j} d_{i,j},$$

subject to

$$\sum_{i=1}^{N} d_{i,j} \leq 1, \quad \forall j \in \{1, 2, \ldots, M\}$$

and

$$\sum_{j=1}^{M} d_{i,j} \leq 1, \quad \forall i \in \{1, 2, \ldots, N\},$$

where $\mathcal{F}_{i,j}$ denotes the fitness of assigning sailor i to job j and D is an assignment matrix such that

$$d_{i,j} = \begin{cases} 1 & : \text{ Sailor } i \text{ assigned to job } j \\ 0 & : \text{ otherwise} \end{cases}$$

The fitness measure $\mathcal{F}_{i,j}$ encapsulates all information relevant to determining the desirability of the match. The construction of $\mathcal{F}$ is described in detail later in the paper. The constraints on D ensure that at most one job will be assigned to any sailor and that no job is assigned to multiple sailors. Note that both constraints are inequalities thus allowing for the possibility that a given sailor is not assigned a job.

Over successive two-week intervals, a group of Naval personnel known as detailers manually construct the set of assignments. They must first communicate with the sailor to determine his or her preferences, then with the officer at the post to determine if the sailor is acceptable for that position. This traditional way of assigning jobs is problematic for a number of reasons, preventing the Navy from adopting longer detailing windows which could help allocate resources more effectively.

The Navy is developing a Web Based Marketplace (WBM) for each sailor to connect to through the Web and bid on a pool of jobs for which he or she is qualified. The WBM provides the sailor information about the jobs' suitability and displays metrics of how this job will affect his career progression, current skills, quality of life, pay grade, and other relevant details. It is an attempt to automate the process of the detailers communicating directly with the sailor. The sailor selects a subset of the available jobs that he would like to obtain, and ranks the selected jobs in decreasing order of preference.

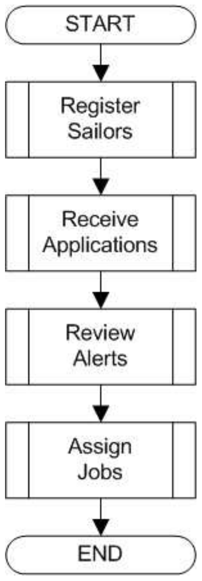

Figure 1: Overall Job Assignment Process

The complete distribution and assignment process is described below. The distribution and assignment process entails three primary phases: allocation, placement, and assignment. The detailing process, the process by which sailors and jobs are actually matched, consists of four steps, as shown in Figure 1.

- Register sailors in the system
- Receive sailors' applications for eligible jobs
- Review of requisitions by the navy command
- Assignment of jobs (match sailor-job)

In the sailor registration step, sailors whose Projected Rotation Date (PRD) is within the specified time range in the application are registered in the database. For each, a set of potential jobs (jobs for which the sailor is qualified) is searched and stored in the database. Additionally, the fitness and cost associated with each job is recorded. The fitness score simply measures the sailor's ability to perform the requirements of the position. The cost measure is known as the Permanent Change of Station (PCS) cost, and estimates the monetary cost of assigning the sailor to that position. The PCS cost includes the cost of moving the sailor and his or her family to the new location, the cost of any additional required training, and many other factors.

After the registration process is completed, the sailors must access the system through a web interface, and apply for the jobs they want. The system allows a sailor to apply for only those jobs for which he is qualified. These applications are stored in the database for use in the detailing process.

The next step, command review, is performed after all applications are received. In this phase, the command reviews the prioritized requisitions (jobs) and checks the suitability of the candidate sailors for each. Sailors particularly well suited to a position may be ranked by officers at the command. In some cases, this stage also provides opportunities for proactive recruiting – contacting sailors who did not apply for the job, but who might be a good fit.

Finally, the matching process takes place during the job assignment phase. Currently, the detailers construct the match entirely by hand, but a prototype system using the Gale-Shapley algorithm described below has been developed. The system produces a set of sailor-to-job assignments that is optimal with respect to the preferences of the sailors and commands. The match is then evaluated by a detailer, who either accepts or rejects the match. Typically, the detailer iteratively refines the set of assignments, rejecting some components of the match for cost or other reasons. If a particular sailor and job pairing is rejected by the detailer, the sailor is re-matched against the list of remaining jobs. The detailer continues this process until all requirements are satisfied. The current state of the art is at best an 80% solution. That is to say, at least 20% of the sailors must be manually assigned by detailers. Obviously, a system which could find suitable assignments for more sailors could provide important benefits for the Navy.

The Navy is focusing on using a Multi-Agent System in solving the Sailor Assignment Problem [9, 11]. The agent system manages and coordinates the applications of sailors for jobs, subsequent evaluation by the command, and finally the construction of the match itself using the Gale-Shapley algorithm. However, the Gale-Shapley algorithm, described below, is not able to consider cost as a factor in determining the best match. A deterministic algorithm, Gale-Shapley considers only the preferences of the sailors and commands. The resulting match is optimal with respect to these preferences, but may be prohibitively expensive to implement. This paper examines genetic algorithm approaches to the sailor assignment problem in order to allow outside constraints such as cost to influence the construction of the match. The goal is to produce matches which keep the sailors and commands happy, while minimizing the total cost of implementing the match.

2. RELATED WORK

2.1 The Gale-Shapley Stable Marriage Algorithm

The Gale-Shapley algorithm is a quadratic time algorithm for finding an optimal set of stable matches, i.e. marriages [7]. To understand the algorithm, let us first simplify the problem to one of finding stable marriages between a set of n men and a set of n women. This algorithm works by allowing either the men or the women to propose. Without loss of generality, let us assume that the men propose to the women. Each iteration of the algorithm, each man proposes to the woman to whom he would most like to marry and to whom he has not previously proposed to. After all the men propose, each woman rejects all but her favorite from among the men who proposed to her. She does not accept the remaining proposal. Instead, she keeps him around in case no one better comes along. After all women have been proposed to at least once, the algorithm terminates and the women each accept the proposal from the man they most prefer. This algorithm produces after $O(n^2)$ steps a set of marriages that is stable and optimal with respect to the preferences of the men. Allowing the women to propose produces a stable set of marriages optimal with respect to the preferences of the women. Note that allowing the women to propose will not generally lead to the same set of marriages as if the men proposed. In fact, this will occur only if there is but a single optimal set of marriages.

The algorithm described above is currently used in many situations in which a stable set of assignments is required. However, the algorithm provides optimality only with respect to local conditions. If the men propose, then each man will be at least as well off under the solution found as under any other stable set of marriages. There is no mechanism by which the global characteristics of the problem can be considered. For the marriage problem, this is a reasonable restriction. However, for other types of assignment problems, this restriction may be problematic.

The Sailor Assignment Problem has two major constraints outside of those imposed in the mathematical formulation of the problem described earlier. The first of these is that a sailor must only be matched to a job for which he meets the requirements. These requirements encapsulate not only the ability (requirements and qualifications) of the sailor to do the job, but also the desirability of offering that assignment from the Navy's point of view. If an experienced, highly paid sailor wished to apply for a job as a cook, the Navy would likely disallow the assignment. The Navy satisfies this constraint by disallowing applications from a sailor to a job he is unqualified for. The system must respect this constraint and never assign an unfit sailor to any job.

A second constraint on feasible solutions is that the PCS cost associated with implementing the match should be minimized. If cost were not an issue, then the Gale-Shapely algorithm could provide an optimal set of assignments with respect to the preferences of the sailors and commands, although such a match would not necessarily involve all sailors. However, the addition of cost constraints results in an NP-hard optimization problem.

2.2 Genetic Algorithms

Genetic Algorithms are a biologically inspired method that have been successfully applied to many real world applications. They have often been utilized to solve the Quadratic Assignment Problem (QAP) with mixed results. Typically, canonical GAs have not fared well on QAP [2]. Better results have been reported by combining GAs with local improvement heuristics such as Tabu Search [12, 6].

However, there are substantial differences between QAP and the Sailor Assignment Problem (SAP). In the standard QAP formulation, it is assumed that every possible assignment from members of one group to another is valid. Stated another way, the standard QAP is unconstrained. In contrast, valid assignments in the sailor assignment problem are determined by a long list of constraints. The Navy's guidelines dictate that only certain jobs are acceptable for any given sailor, and any system to produce these assignments must respect those guidelines.

The constraints on valid assignments has profound consequences for the design of a genetic algorithm to solve the problem. In QAP, the most commonly used representation is the permutation. In a typical QAP problem, the number of facilities is equal to the number of locations, thus restricting solutions to one-to-one mappings from the set of facilities to the set of locations. Because of this, a permutation of the numbers 0 through $n-1$ always defines a valid set of assignments. In the sailor assignment problem, there are typically more jobs than sailors at any given time. In addition, a given sailor can select only a subset of the available jobs due to lack of training, differences in pay grade between the sailor and the proposed job, or many other factors. As a result, only a very small fraction of randomly generated permutations would correspond to valid assignments. This implies that most genetic operators commonly used on permutation encodings would be ineffective.

For the same reasons, a binary encoding is unsuitable for the sailor assignment problem. A mutation operator which simply flipped a bit would likely produce an infeasible individual, as would crossover applied at arbitrary points on the string. Instead, we utilize an integer encoding in which each gene is chosen from a subset of possible numbers, precisely those representing valid jobs for that sailor.

Hybrid genetic algorithms have also been applied to the Generalized Assignment Problem (GAP) with great success [3, 5]. In GAP, there are n agents and m jobs to be assigned and the goal is to find an assignment of each of the jobs to one agent such that no agent violates his capacity constraints. Chu and Beasley [3] proposed a hybrid steady-state genetic algorithm for GAP which was able to surpass the state of the art on large and difficult instances. One of the key aspects of the their work was the development of a replacement scheme which always replaced the most infeasible individual from the population. If all members of the population were feasible, the method simply replaced the individual with the lowest fitness. Their algorithm was subsequently modified providing improvements in both execution time and final solution quality [5]. One modification was to replace the most-infeasible-first replacement strategy with a more traditional penalization mechanism.

The sailor assignment problem can in principle be formulated as a GAP in which each job requires one capacity unit and each agent (sailor) has only one capacity unit to utilize. In doing so, most of the assumptions that make the preceding algorithms effective are violated. The most general formulation of the problem assumes that persons can have more than one job; that assumption is pervasive in the algorithms. Thus, the GAP oriented genetic algorithms cannot achieve good performance if directly applied to SAP. Stated another way, if the number of jobs is greater than the number of sailors, not all jobs can be assigned. This point underlies the conceptual difference between the GAP and the SAP. Because our algorithm must perform well in the presence of many unassigned sailors, the GA approaches to the generalized assignment problem discussed above are problematic.

In this paper, we examine alternative genetic algorithm approaches to the sailor assignment problem in order to allow outside constraints to influence the quality of the match. The goal is to produce matches which keep the sailors and commands happy, while minimizing the total cost of implementing the match. Additionally, we examine the ability of the GA to generate multiple good solutions to SAP through niching techniques.

3. IMPLEMENTATION

One of the primary decisions to be made concerning any genetic algorithm is the encoding. For the Sailor Assignment Problem, candidate solutions are represented using an integer encoding scheme specific to the application. Each chromosome is encoded as an array of integers, where each position in the array represents a particular sailor and the number stored at that position represents the job assigned to that sailor. To ensure that no sailor receives a job for which is not fit, each sailor is associated with a set of jobs that he is allowed to take. All operators used by our genetic algorithm ensure that no position in the array can take on a value not in this set.

This encoding ensures that no sailor gets a job for which he is not qualified, but does so at the expense of allowing another type of constraint to be violated; a single job may be assigned to multiple sailors. Whenever this occurs, a heuristic is used to attempt to repair the individual. For any sailor in the match with a job already awarded to another sailor, a list of each job the sailor may take is generated in random order. The list is traversed looking for the first job which is not currently assigned to another sailor. If no job can be found satisfying this criteria, then the sailor is left unassigned in the match. Unassigned sailors contribute nothing to either the match fitness score or to the PCS cost of the match, and are indicated with a value of -1 in the encoding. In order to eliminate bias resulting from position in the encoding, the order in which sailors are visited is randomized prior to beginning the repair process.

In addition to the repair mechanism, a heuristic was employed to generate the initial population. Each sailor is visited in turn and assigned a random job which has not been assigned to another sailor. If no such job exists, the same heuristic repair process described above is utilized. To prevent bias based on the order of the sailors and jobs, both the order in which the sailors are visited and the order in which the jobs are attempted are randomized prior to initialization of the individual.

Providing a solution in which some sailors have not been assigned is appropriate for the application, provided that the number of such sailors is relatively small. It may be that no feasible assignment of all sailors exists. Furthermore, it may be that although a complete match exists, it is far more expensive than another set of assignments which does not match all sailors. In that case, it may be better from the Navy's viewpoint to manually match the remaining sailors by relaxing constraints or assigning additional training.

Although unmatched sailors are acceptable in small numbers, it is obviously preferable to find a job for as many sailors as possible. Crossover and mutation operations may produce individuals with unnecessarily unassigned sailors. Consider a solution in which two sailors are assigned the same job. The repair procedure described above, if it fails to find an alternate job for one of the sailors, will leave one unassigned. Later, crossover or mutation may alter the other sailors in such a way as to remove the potential conflict. Therefore, the repair procedure also attempts to provide a valid assignment for any sailors previously marked as unassigned. In addition, a small penalty is applied for each unassigned sailor in a match. The penalty is kept reasonably small to prevent the GA from focusing entirely on matching all sailors at the expense of fitness and cost.

Several variants of genetic algorithms were tested. The results are reported for the most promising parameter settings. Both generational and steady state GAs using Stochastic Universal Sampling (SUS) selection [1] and uniform crossover [14] were tested. The steady state GA is elitist in nature, always replacing the worst individual in the parent population. For the generational GA, we tested both elitist and non-elitist algorithms. In addition, tournament selection and both one and two-point crossover were tested for each algorithm. Uniform crossover was found to be produce better results for all but the smallest problem.

In addition to the two conventional GAs mentioned above, a version of CHC [4] was tested. CHC is a state-of-the-art genetic algorithm which uses a sophisticated convergence detection mechanism to trigger a cataclysmic mutation of the population. CHC's crossover operator, called HUX, implements an incest prevention mechanism with a dynamically changing definition of incest. During each generation, the algorithm randomly pairs individuals for recombination. The selection is done without replacement, so each individual is selected once. The individuals are then allowed to produce offspring only if they are not too similar. Thus, the number of children produced per generation is dynamic. When no children are produced during one generation, the similarity threshold is reduced, allowing more similar individuals to successfully breed. When this threshold reaches zero, the cataclysmic mutation is triggered. This creates a new population containing exactly n copies of the best solution in the old population, where n is the population size. One copy is kept unchanged, while the remainder of the new population is constructed by randomly flipping 35% of the bits in the remaining copies.

The algorithm was modified only to handle the constrained integer encoding scheme rather than the conventional bit-string encoding. The sole difference is that the "Hamming distance" for the SAP is defined to be the number of sailors assigned different jobs by two solutions. When convergence was detected, the restart was performed and 35% of the sailors in each solution were assigned randomly selected jobs.

The repair procedure was then invoked to handle constraint violations in the newly created individuals.

All runs of the GAs except CHC used a population size of 300 and were terminated after 500,000 evaluations of the fitness function, or in the case of elitist algorithms, if the population had been converged for some fixed period of time. Following the literature, CHC was tested with a population size of 50.

4. EXPERIMENTS AND RESULTS

Data was gathered from actual Navy requisition data for data sets of 20 sailors to 533 sailors. To generate the data, randomly selected subsets of sailors were drawn from the database. Any job that a selected sailor was eligible for was also selected from the database. The training score and PCS cost of each of these candidates was computed and stored for use in the fitness function evaluation. Because there is a limited range of data which can be taken from the database, a random problem generator was also employed to provide additional test instances of various sizes. This generator accurately models the distribution of sailors and jobs present in real situations. Each synthetically generated sailor has approximately the same number of eligible jobs and applies for roughly the same number as one could expect to occur in reality.

To construct random problem instances, the distribution of data drawn from the database was examined. The generator was then constructed to match this distribution. Parameters of the generator include the total number of sailors in the window, the total number of jobs in the window, and the mean and standard deviation of the number of jobs applied for by each sailor. The training scores and PCS costs were drawn from a distribution constructed to be identical to that found in the real-world data.

Table 1 shows the number of sailors and jobs in each instance of the problem used during testing. The *a* versions of the problems are drawn from actual sailor data. The 20, 50, 100, and 200 sailor versions of the problem are constructed by taking a random sample of the sailors eligible for reassignment during a given time window and selecting all jobs for which the selected sailors are eligible to apply. The 533 sailor instance represents all sailors and jobs eligible for reassignment during a particular two-month window. Note that the number of jobs is nearly independent of the number of sailors. The implications of this are that competition for jobs increases substantially as the number of sailors increase, so that not only does the dimensionality increase, but also the difficulty associated with finding a feasible solution is increased.

The amount of real data is limited, and many of the *a*-type problems exhibit a high degree of overlap among the sailors and jobs. Therefore, we focus primarily on the synthetically generated instances of the problem. This lets us avoid any bias associated with the particular set of sailors and jobs present in the database. In most experiments, we report only the full 533-sailor instance to analyze the performance on the actual sailor data.

Each sailor is allowed to apply for only a subset of the jobs available. The exact jobs available to a sailor depend on several factors defined by the Navy. For each job a sailor is allowed to apply for, a measure of utility is computed encapsulating information such as discrepancies between the pay grade of the sailor and the job, the training requirements for the job, and several other factors. We can define a $n \times m$ matrix A where $a_{i,j}$ is the utility associated with assigning sailor i to job j. This score is also referred to as the total training score and is defined by

Table 1: Number of sailors and jobs in each instance of the problem considered. The "a" instances consist of real data. The "b" instances are created using a random problem generator.

Problem	Sailors	Jobs	Problem	Sailors	Jobs
20a	20	1002	50b	50	750
50a	50	1127	100b	100	1500
100a	100	1663	200b	200	1500
200a	200	1676	1000b	1000	2000
533a	533	1582			

$$T = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{i,j} d_{i,j}. \quad (1)$$

To begin, we first establish a baseline of comparison by running the Gale-Shapley algorithm on each problem instance. The resulting match is optimal with respect to the preferences of the sailors. Without loss of generality, we assume that each sailor applies for all jobs for which he is eligible to apply, and that he does so in the order of decreasing fitness. As a result, the match is not only optimal for the sailors, but also with respect to the total fitness. Certainly, the GA will not be able to consistently match Gale-Shapley on this set of problems.

The implications of these assumptions are that the Gale-Shapley algorithm will provide the match with the highest total fitness score. In a real scenario, this is not necessarily true. The algorithm will provide the match which maximizes the overall satisfaction of the sailors and commands, subject to the hard constraints built into the problem. However, any solution inside the feasible region of the space will be considered equivalent with respect to the constraints. Therefore, the algorithm cannot consider the impact of outside influences on the problem. If the set of sailor preferences diverge from the assumed preference for higher fitness, then the match scores will necessarily decrease for the Gale-Shapley algorithm.

Because there exists a very fast algorithm for finding optimal matches based on the preferences of the sailors, there is little to be gained from running a genetic algorithm on this formulation of the problem. However, as stated above, there is no way for the Gale-Shapley algorithm to incorporate external constraints. This represents a substantial problem for the Navy, since it prevents the simultaneous minimization of the PCS cost of the match. Therefore, it is important to show that the GA is capable of producing not only good matches, but also matches which result in smaller cost to the Navy.

Ten trials of the genetic algorithms were then performed on the same problem instances. Table 2 clearly shows the advantage of the Gale-Shapley algorithm when only the fitness relationship between sailors and jobs is considered. While CHC is able to outperform the steady state and generational GAs on all instances, it is never able to fully reach the performance of the deterministic algorithm.

To determine the ability of the GA to minimize the PCS cost while maintaining highly fit matches, the fitness calcu-

Table 2: Comparison of Gale-Shapley and the Genetic Algorithms using only the fitness of the individuals to construct the match.

Algorithm	20a	50a	100a	200a	533a
Gale-Shapley	15.36	37.40	75.22	148.26	475.07
Generational GA	14.97	35.06	60.78	115.25	447.44
Steady State GA	14.75	36.06	72.73	143.26	459.91
CHC	15.10	36.65	74.35	146.39	468.84

Table 3: Comparison of Gale-Shapley and the Genetic Algorithms on problems using both fitness and PCS Cost. For these experiments, fitness and PCS cost are equally weighted.

Problem	CHC		Generational GA		Steady State GA		Gale-Shapley	
	Fitness	PCS Cost	Fitness	PCS Cost	Fitness	PCS Cost	Fitness	PCS Cost
50b	44.20	370,229	43.84	373,966	44.05	371,816	48.20	1,218,602
100b	93.16	448,460	91.28	606,313	91.79	552,413	98.47	2,313,229
200b	183.55	897,919	170.12	2,294,836	180.45	1,229,460	195.61	4,999,665
533a	463.32	4,340,020	445.19	4,695,708	457.73	4,453,584	475.07	4,566,906
1000b	870.77	9,493,592	781.65	21,837,280	858.37	11,531,480	936.71	25,624,575

lation of the GA was modified to include a PCS cost term. The new fitness measure is a linear combination of match fitness and PCS cost. Because the goal is to minimize the PCS cost while maximizing the total fitness given by Equation 1, an inhibitory factor is used for the PCS component of the fitness function. Both the match fitness and PCS cost must be normalized prior to computing the fitness of a match. Because the sailor-to-job fitness scores are already normalized values, they are simply averaged over the entire match to yield the match fitness score. For PCS cost, each sailor-to-job cost is divided by the maximum PCS cost for any sailor-to-job candidate in the problem instance. These scaled costs are then averaged over the entire match. The fitness of an individual is then given by

$$\mathcal{F}(x) = \alpha T_x - \beta C_x,$$

where T_x is the scaled total training (fitness) score for match x and C_x is the scaled total PCS cost for match x. The normalized PCS costs are typically less than the normalized fitness scores, due to the presence of possible matches with very large PCS costs. These matches are quickly rejected by the GA, leading to a population in which most of the individual matches consist of relatively low cost compared to the average fitness score.

Table 3 shows the performance of the algorithms with $\alpha = \beta = 1.0$. Note that the Gale-Shapley match does not utilize the PCS cost in determining its match. Thus, while the fitness score of the Gale-Shapley match is superior to those found by the genetic algorithms, this comes at the expense of a much more costly match. Both variants of GA were able to find matches exhibiting much smaller PCS costs than the Gale-Shapley match, but CHC was consistently better than the steady state or generational algorithms in that regard.

Consider the performance on the 100 sailor problem from Table 3. We see that the Gale-Shapley algorithm produced a match with a total match fitness of 98.47 with a PCS cost of \$2,313,229.49. Over all 100 sailors, the mean fitness and cost are 0.985 and \$23,132.29. Contrast those numbers with those reported by the CHC algorithm. The mean fitness for the sailors in that match is slightly lower, 0.932. However, the average cost per sailor is only \$4,484.60, approximately one-fifth the cost of the match found by the Gale-Shapley algorithm.

Because the GA uses a simple linear combination of its individual components to evaluate fitness scores, it is relatively easy to guide the algorithm toward better matches or lower costs as the need becomes apparent. For example, the PCS costs reported in Table 3 are so small that it might be worthwhile to experiment with increasing the weight associated with high fitness. In this way, it is hoped that the GA will find better matches at the expense of slight increases in PCS cost. Table 4 shows the results of repeating the algorithms using weights of $\alpha = 1.0$ and $\beta = 0.5$. This setting allows the GA to more aggressively improve the fitness of the match, since any penalty incurred by an increase in PCS cost is reduced by a factor of two. The results show that the GA can indeed find increasingly better match scores without dramatically increasing the PCS costs.

Both with and without consideration of the PCS cost, the problem `533a` is significantly more difficult for the GA to optimize. While CHC is capable of finding good solutions costing less than those found by the Gale-Shapley algorithm, the difference is much less than reported in the other problems. The difficulty of this problem arises from the large number of sailors applying for each job – a direct result of the assumption that all sailors would apply for each job they were eligible to receive. The mean number of sailors per job in this problem is 22.68. Compare this to 5.05 sailors per job in the instance `1000b`. Intuitively, the more sailors apply for each job, the more likely it becomes that an arbitrary candidate solution assigns multiple sailors the same job, thus incurring a penalty which prevents the individual from contributing to the evolutionary process.

4.1 Finding Multiple Solutions

The introduction of PCS cost into the fitness function transforms the problem into a multiobjective optimization problem. In many multiobjective optimization problems, the goal is not only to find an optimal solution, but to uniformly sample the Pareto front, the set of equivalent but distinct optimal solutions.

For the Navy's Sailor Assignment Problem, often there is no clearly preferred solution. Instead, there are a range

Table 4: Comparison of Gale-Shapley and the Genetic Algorithms on problems using both fitness and PCS Cost. For these experiments, fitness is weighted more heavily than PCS cost.

Problem	CHC		Generational GA		Steady State GA		Gale-Shapley	
	Fitness	PCS Cost	Fitness	PCS Cost	Fitness	PCS Cost	Fitness	PCS Cost
50b	45.40	460,292	45.81	522,127	45.57	472,132	48.20	1,218,602
100b	94.36	501,393	93.75	696,454	93.34	643,305	98.47	2,313,229
200b	187.75	1,257,060	178.84	2,750,758	185.18	1,658,778	195.61	4,999,665
533a	464.19	4,491,036	446.42	4,768,400	456.35	4,529,148	475.07	4,566,906
1000b	908.71	12,218,340	773.17	23,757,540	894.22	14,329,880	936.71	25,624,575

of solutions across the Pareto front from which a human detailer must choose. The deterministic nature of the algorithm and lack of side constraints prevents the Gale-Shapley formulation from finding more than a single solution. The GA however, should be able to present multiple solutions to the detailer, allowing greater flexibility in the assignment process. The typical GA approach to Pareto optimization is the introduction of niching techniques.

Niching Genetic Algorithms (NGAs) attempt to bias the GA away from areas which are currently overrepresented in the population. The most common means of introducing this bias is Goldberg and Richardson's [8] fitness sharing scheme. In this approach, individuals undergo a fitness scaling operation just prior to selection. The fitness of each individual is reduced proportionally to the number of other individuals in the same region of the space. To determine whether two individuals should be considered to be in the same region, a single parameter, σ_{share}, is specified. This parameter defines the maximum distance between two individuals considered to share the same niche. The selection algorithm then sees only the scaled fitness values. By manipulating the scaled values, it is possible to manipulate the probability of survival for a given individual, thus providing some means of control over which regions of the space are explored further.

To test the ability of the GA to find multiple good solutions to the assignment problem, we performed the experiments using fitness sharing and recorded the best individual found from each niche. In accordance with various other studies [15, 13], we used SUS selection with one-point crossover in order to increase the likelihood of convergence. Additionally, following [15], we terminated the GA if 20 generations passed without significant improvement in the mean fitness of the population. Also to promote convergence, a reduced crossover probability of 0.7 was tested along with a higher probability of 0.9. Under these parameter settings, the niching GA failed to converge, producing approximately n very sparsely populated subpopulations, where n is the population size.

To attempt to allow the GA to converge, we then employed an elitist replacement strategy in which the best individuals always survive to the next generation. With this modification, the sharing mechanism provided distinct subpopulations, but failed to maintain them throughout the entirety of the run. Therefore, we modified the termination criteria to halt after the number of subpopulations decreases to a fixed level – five in our experiments. We then report the most fit individual from each remaining subpopulation for the purpose of human evaluation. Table 5 shows the results of these experiments.

The interpretation of σ_{share} in a typical parameter optimization NGA is that it should be very close to the distance between any two peaks in the fitness landscape. If the value of σ_{share} does not match the structure of the fitness landscape, then the ability of the NGA to adequately find and maintain multiple solutions will be negatively affected. In the sailor assignment problem, σ_{share} plays the same role. However, the distance measure between any two individuals is defined simply as the number of sailors assigned different jobs by the two candidate solutions.

We tested several different values for σ_{share} with the fitness sharing method. The behavior of the NGA was essentially independent of σ_{share} for reasonable settings. With generational replacement, the GA failed to converge, while converging completely with elitist replacement. For the experiments reported, we used $\sigma_{share} = 0.15n$, where n is the number of sailors for the problem.

Thus, the interpretation of Table 5 is as follows. The GA finds approximately five distinct matches such that no two of the five assign more than $N - \sigma_{share}$ sailors to the same job. For example, consider the problem `200b`. There are different assignments of the 200 sailors to 200 jobs found during each run of the GA. Any one of those five assignments may be compared to any of the other four and at least 50 sailors will be assigned to a different job in the two matches.

For comparative purposes, we then tested Mahfoud's [10] Deterministic Crowding algorithm. Because DC includes its own selection and replacement operators, the vital parameters are the population size and the crossover operator. One-point crossover was selected due to lower disruption, although the elitist nature of the Deterministic Crowding algorithm should allow for convergence with more aggressive genetic operators.

Like the sharing GA with elitism, the Deterministic Crowding GA converged but was unable to prevent complete convergence. Therefore, we employed the same early termination criteria to halt the GA before all subpopulations were lost. Table 6 presents the results of these experiments with the Deterministic Crowding algorithm. To determine the number of distinct solutions found, the final population was sorted and the best solutions which assigned at least 15% of the sailors different jobs were reported.

As shown in Table 5, the matches are typically of relatively high fitness. This implies that there are multiple very different solutions which may satisfy the Navy's requirements for both fitness and cost. Thus, the ability to present multiple possible solutions to a human detailer could be very valuable, as it allows the detailer to consider factors which may not be well represented in the fitness function. While this ability is useful, it is important to limit the information presented to the detailer. Presenting thousands of possible

Table 5: Performance of the GA using fitness sharing and early termination on the task of finding multiple good solutions.

Problem	Solutions	Fitness	PCS Cost
50b	4.7	45.36	480,348
100b	4.2	94.14	602,136
200b	4.5	186.35	1,464,456
1000b	4.8	872.20	21,190,634
533a	4.4	448.86	4,678,452

Table 6: Performance of the GA using deterministic crowding with early termination on the task of finding multiple good solutions.

Problem	Solutions	Fitness	PCS Cost
50b	3.9	45.64	480,919
100b	4.6	94.36	568,905
200b	4.3	186.05	1,568,144
1000b	4.4	883.61	15,943,000
533a	4.7	451.33	4,897,398

matches is typically far worse than presenting one. Any information presented to the detailer must be in a form that allows the detailer to use and process the information. The ability to specify a maximum number of matches is thus a useful trait of our implementation.

5. CONCLUSIONS

We have shown that a genetic algorithm has tremendous potential to improve the Navy's ability to generate sailor-to-job assignments in a much more cost effective way than is currently in use. In addition, our algorithm is shown to possess the ability to present multiple different solutions to a human detailer, thereby allowing a greater degree of human control over the detailing process without subjecting the detailer to an avalanche of information.

The failure of the unmodified fitness sharing mechanism to converge to a set of stable subpopulations suggests an area in which our approach could be improved. Termination of the algorithm when a specified number of solutions is reached can be a useful tool, as described above. A more stable niching procedure could allow the individuals in each subpopulation to further evolve, perhaps finding better solutions. An area of future work thus involves studying how best to find and maintain multiple different solutions to the assignment problem.

Additionally, on the most difficult instance of the problem considered here, `533a`, the enormous number of job applications for each sailor results in markedly decreased ability to generate new individuals without large numbers of constraint violations. This dramatically reduces the effectiveness of the GA, although CHC with its restart mechanism is still able to find very good solutions less costly than the Gale-Shapley match. The addition of an hybridization mechanism has the potential to provide greater improvements on these most difficult instances of the problem. It should be noted, however that although instance `533a` was drawn from actual fitness and cost data, it is a pathological case in that typically sailors would not apply for all of the potentially thousands of jobs available to them.

6. ACKNOWLEDGMENTS

This work was supported by a grant from Naval Personnel Research, Studies, and Technology (NPRST) through Battelle Scientific Services Program (Task #TCN04157).

7. REFERENCES

[1] J. E. Baker. Reducing Bias and Inefficiency in the Selection Algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21, 1987.

[2] E. Cela. *The Quadratic Assignment Problem: Theory and Applications.* Kluwer Academic Publishers, 1998.

[3] P. C. Chu and J. E. Beasley. A Genetic Algorithm for the Generalized Assignment Problem. *Computers and Operations Research*, 24(1):17–23, 1997.

[4] L. Eshelman. The CHC Adaptive Search Algorithm. In *Foundations of Genetic Algorithms I*, pages 265–283. Morgan Kaufmann, 1991.

[5] H. Feltl and G. Raidl. An Improved Hybrid Genetic Algorithm for the Generalized Assignment Problem. pages 990–995, 2004.

[6] C. Fleurent and J. Ferland. Genetic Hybrids for the Quadratic Assignment Problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 16, Quadratic Assignment and Related Problems:173–187, 1994.

[7] D. Gale and L. S. Shapley. College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[8] D. E. Goldberg and J. Richardson. Genetic Algorithms with Sharing for Multimodal Function Optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, 1987.

[9] J. Liebowitz and J. Simien. Computational Efficiencies for Multi-Agents: A Look at the NPRST's Multi-Agent System for Sailor Assignment. 2004.

[10] S. W. Mahfoud. A comparison of parallel and sequential niching methods. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 136–143. Morgan-Kaufmann, 1995.

[11] L. McCauley and S. Franklin. A Large-Scale Multi-Agent System for Navy Personnel Distribution. *Connection Science*, 14(4):371–385, December 2002.

[12] P. Merz and B. Friesleben. A Genetic Local Search Approach to the Quadratic Assignment Problem. In *International Conference on Genetic Algorithms (ICGA '97)*, pages 465–472, 1997.

[13] B. Sareni and L. Kr ahenb uhl. Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation*, 2(3), September 1998.

[14] G. Sywerda. Uniform Crossover in Genetic Algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9, 1989.

[15] J.-P. Watson. A Performance Assessment of Modern Niching Methods for Parameter Optimization Problems. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, 1999.