Proceedings of the 1996 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems

Maximizing Multi-Sensor System Dependability

R. R. Brooks
Communications Science and Technology Center
California State University
100 Campus Center/Building 18
Seaside, California 93955-8001, USA

S. S. Iyengar
Department of Computer Science
Louisiana State University
Baton Rouge, Louisiana 70803-4020, USA

Keywords

Dependability, Distributed Agreement, Sensor fusion

Abstract

This paper considers distributed sensor systems and finds redundant configurations which maximize dependability while insuring the system remains within cost or weight constraints. Given different sensor modules which fulfill the system's operational requirements but have different dependability and cost parameters, efficient methods are used to find maximum dependability configurations. These methods limit the search to a constrained subspace of the problem space. It is shown that this region must contain the optimal configuration. Three heuristics: genetic algorithms, simulated annealing and tabu search are used. Experimental results are presented with dependability gains of between 10 and 15%. These test cases compare results from all methods and verify that in most cases the simulated annealing heuristic provides the best solutions.

1 Introduction

The past decade has seen an explosive growth in the study of multi-sensor fusion and integration for intelligent systems applications. Sensor integration increases the ability of automated systems to interact with their environment by synergistically combining readings of independent sensors into logical representations [17,21]. The use of highly redundant sensing systems is one of the primary areas included in sensor integration; methods not based on redundancy have been found particularly sensitive to sensor noise [24]. Redundancy increases dependability, but also weight and cost. Success in designing redundant systems depends on making the best possible trade-off at least cost.

Systems using redundant sensors exist in several key areas. Defense applications include ALVs (Autonomous Land Vehicles), intelligent pilot's associates, and battle management systems [21]. Intelligent manufacturing uses distributed components with independent sensors [17]. Surveillance systems use distributed sensors and intelligent components for military applications like Desert Storm , and monitoring borders to stop drug smuggling.

The goals for integrating sensor systems are: (1) to allow for sensor variation in reliability, noise and error, (2) to accommodate a large number of sensors achieving full sensor integration, (3) to isolate heterogeneity in low level modules, (4) to enhance flexibility; and (5) to maintain a common representation of the environment [27]. Highly redundant sensor systems have several advantages:

- Multiple inaccurate sensors can cost less than a few accurate sensors [1].
- Sensor reliability can be increased [1,27].
- Sensor efficiency and performance can be enhanced [27].
- Self-calibration can be attained [27].
- More flexible sensor architectures are possible [1,27].

Current problems are [27]:

- Fusing redundant and multi-sensor readings [5,6,10,17].
- Methods for self-calibration [7,9].
- Architectures for improving sensor efficiency[17].
- Selecting sensors to improve reliability and resolve resource allocation conflicts. An approach using system cost and sensor accuracy is in [1]. One minimizing cost is in [11]. This paper presents a methodology which considers cost or weight constraints and maximizes dependability within those constraints.

Our approach assumes standardized components allow redundancy among heterogeneous components. Good engineering practice dictates the use of commercial off-the-shelf (*COTS*) components to reduce system complexity and cost, especially for redundant systems [28]. A sensor interface standard from NIST allows for the use of heterogeneous components without effecting system complexity [22]. For sensor systems, the use of heterogeneous hardware is innately beneficial to system dependability [18].

N-modular redundancy (*NMR*) uses results from *N* components in parallel which can be compared to insure correctness [25]. *NMR* systems may infer correct information in the presence of failures in a relatively large number of modules, this is the Byzantine Generals Problem (*BGP*) [20]. *BGP* algorithms make a unanimous decision in the presence of arbitrary errors when less than one third of the modules are faulty [2]. A similar problem exists for sensor fusion applications. The relationship between sensor fusion and the Byzantine Generals problems is explained in [8]. The sensor fusion and *BGP* extensions to *NMR* aid in the design of reliable systems and guarantee that a system will function correctly as long as more than half (in the case of one-

0-7803-3700-X/96 $5.00

dimensional sensor fusion) or more than two-thirds (in the case of Byzantine Agreement) of the components function correctly.

We assume component failure is statistically independent. The dependability statistics for components may be based on several models. For sensor systems, dependability models reflect the problem considered; if the design is concerned with mechanical failure then mean-time-to-failure and mean-time-to-repair are adequate. For transient and intermittent errors, the distributions of fault arrival and duration must be known. Our examples use exponential distributions for component failure and repair for tractability and consistency with reliability literature. Equations are derived for systems where over half the components must be functional, this can be changed by replacing *N/2* as necessary.

This paper is organized as follows. Section two presents metrics for redundant system dependability. The problem space for our design problem is presented in section 3, along with description of the subspace which must contain the optimal choice. Section 4 presents three heuristics used for this problem: genetic algorithms, simulated annealing, and tabu search. Results from a sample problem is given in section 5. Section 6 presents our conclusions.

2 Dependability Measure of Redundant Systems

Dependability is used in a generic sense to address either reliability or availability. Reliability is the probability that a system is functional at the mission time (*T*). It is used for single mission systems where repair is unfeasible. Availability is the percentage of time a system will be functional when the system reaches a steady state. Availability is used for systems which can be maintained and repaired. We describe a technique to measure dependability for *NMR* sensor systems.

To compute reliability, we either use a Markov model or perform a combinatorial analysis. Figure 2 shows a Markov chain modeling system dependability (for reliability the repair rate μ is zero). A set of differential equations can be derived directly from the diagram, and solved using Laplace transforms or numerically. The combinatorial analysis assumes each component has an identical probability of success $r(t)$ ($= e^{-\lambda t}$, if a component has a constant failure rate λ). Let $q(t) = 1 - r(t)$. The assumption of statistical independence allows us to use Bernoulli's law, which finds the probability of i out of N components working at time t as:

$$\binom{N}{i} [r(t)]^i \; [q(t)]^{N-i} \qquad (1)$$

The reliability for the system is the summation of the terms with i varying from N to $\lfloor N/2 \rfloor + 1$. Both approaches derive the same answer. The combinatorial approach has two advantages: 1) It is independent of the distribution defined by

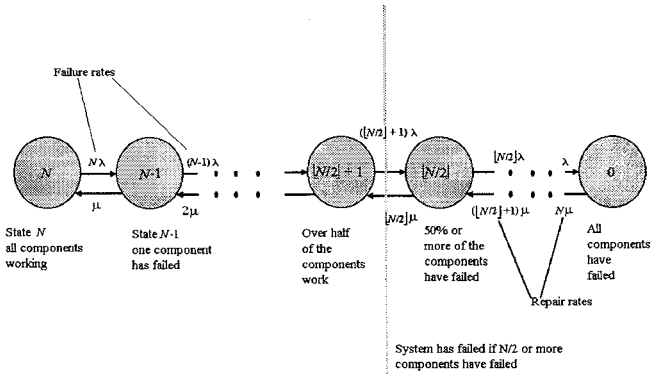

Figure 2. Markov chain of availability model for a system which can tolerate failure of up to half its components.

the reliability function $r(t)$. 2) It is easier to apply when more than one component type is used.

Consider a system with two component types. If $N_1(N_2)$ is the number of components of type 1(2) the derivation using Bernoulli's law can be used to cover the new configuration. Consider the cases where no components of type 1 have failed, one component of type 1 has failed, *etc.*, up to the case where all N_1 components of type 1 have failed. These cases are disjoint and the sum of their probabilities is one [12]. It partitions the sample space giving an expression for system reliability at time t in terms of $r_1(t)$ the reliability of component type 1, and $r_2(t)$ the reliability of type 2.

$$R(t) = \sum_{k=0}^{N_1} \binom{N_1}{k} [r_1(t)]^k [1-r_1(t)]^{N_1-k} \left[\sum_{m=\lfloor N/2 \rfloor + 1-k}^{N_2} \binom{N_2}{m} [r_2(t)]^m [1-r_2(t)]^{N_2-m} \right] \qquad (2)$$

The concept can be extended to more component types. Evaluating a combination of J different types of components requires J levels of summations in the format of (2). An efficient algorithm for this is given in [11].

For availability the repair rate μ in Figure 2 is non-zero. Since the chain is finite and strongly connected, the system represented by the model will reach a steady state. The formula for availability is the same as reliability equation (1), except component reliability $r(t)$ is replaced by component availability $a(t)$. For components with a constant failure rate λ and a constant repair rate μ, the steady state component availability a will be $\mu/(\mu+\lambda)$, and steady state component unavailability is $1-a$ or $\lambda/(\mu+\lambda)$. Equation (2) and the algorithm in [11] can also be used if component availability replaces component reliability.

Component dependability is constrained to a range of values between 0 and 1. A value of 1(0) indicates a component never fails(functions). The dependability value of a system tends towards zero, remains about 1/2, or asymptotically approaches 1 with the increase in the number of components comprising the system, provided the dependability measure of the components is less than 1/2, exactly 1/2, or greater than 1/2, respectively. It, thus, depicts the "S-Shaped property" described in [3]. If components are

perfect, system reliability will be 1 no matter how many components are used. Fault masking systems are feasible for components with dependability values are between 50% and 100%. Only dependability values within this range should be considered. This paper implicitly assumes components fit this requirement As long as the system dependability constraint is less than 1 and individual component dependability is greater than 1/2, system dependability asymptotically approaches 1 as the number of redundant components increases.

3 Subspace Containing Optimal Choice

A method for determining the dependability of redundant sensor system configurations has been derived in section 2. Here, we find a model for the most dependable configuration that fulfills cost (or weight) constraints. Components have either known component reliability (for systems with a fixed mission time) or known component availability (for systems where the percentage of the time the system is functioning is most important.) Each component type is characterized by dependability statistics and positive per item cost.

Cost refers to a limiting factor. Most often this is a dollar amount. It may also be weight, power consumption, bandwidth or a number of factors. The method presented can be directly applied in any of these cases.

We consider each combination of J component types as a point in a discrete J-dimensional space. The point is described by a J-dimensional vector $(x_1,x_2,...,x_J)$ where each x_i corresponds to the number of components of type i in the system. If the choice is to be made among three types of components, the combination of 2 components of type 1, 25 components of type 2, and none of type 3 corresponds to point (2, 25, 0). Since each type of component has a given per item cost it is possible to determine the cost of each combination. If component type i has cost c_i, the cost of combination (2,25,0) is $2*c_1 + 25*c_2 + 0*c_3$. The system cost is:

$$\sum_{i=0}^{J} c_i * x_i \qquad (3)$$

We maximize system dependability using equation (2), within cost bounds given by a maximum value allowed for equation 3. This is a combinatorial optimization problem which can not be solved by mathematical programming techniques like the Simplex algorithm for linear programming, or integer programming. These techniques are inappropriate since equation (2) is non-linear [13].

We must find the optimal point in a J-dimensional solution space where J is the number of component types under consideration. The region with valid solutions is known as the feasible set in the J-dimensional space [26]. When $N/2$ ($N/3$) failures can be tolerated, adding fewer than 2(3) components to configuration C creating configuration C' causes the dependability of C' to be less than the

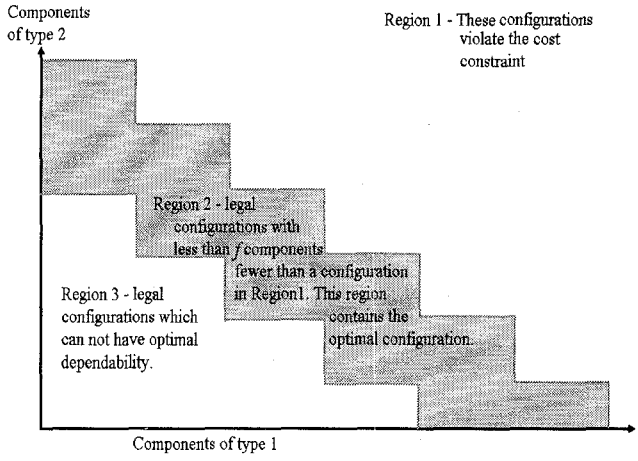

Figure 3. The problem space for a 2 dimensional problem.

dependability of C. C' contains more components which may fail than C but the same number of failures can be tolerated in both C and C'.

Theorem 1- Given J different component types with unit costs c_i. A system is to be designed which tolerates N/f component failures, where N is the total number of components in the system. The configuration which maximizes the dependability measure within cost constraints must have between 1 and f fewer components than a configuration which violates the cost constraint.

Proof - We divide the problem space into three distinct regions: (1) those violating the cost constraint, (2) those which do not violate the cost constraint but have up to f fewer components than a configuration that violates the constraint, and (3) those configurations with more than f components fewer than all configurations violating the constraint.

Configurations in region (1) are excluded by definition. For a configuration C in region (3) with the number of components equal to 1 modulo f, note that adding fewer than f components lowers the configuration dependability. Adding exactly f components of any component type, however, gives a configuration C' which can tolerate one more failure than C and the "S-shaped" property guarantees that C' will have a dependability which is greater than the dependability of C. By applying the same logic to C' it is shown that configuration dependability increases by adding components in quantums of f until the resulting configuration is in region (1). The maximum value must therefore be in region (2). Figure 3 provides an example problem subspace for two components with the regions clearly labeled.◊

4 Heuristics Used

For non-linear combinatorial optimization problem. The problem space has local maxima. To solve problems containing a large number of component types a heuristic is needed to limit the number of configurations considered. We

test three heuristics: genetic algorithms, simulated annealing, and tabu search. We use these variables: J as the number of component types, c_i as the unit cost of component type i, x_i as the number of components of type i in the configuration, and N_i as the maximum number of components of type i possible in a configuration. N_i is $\lfloor$cost limit / $c_i\rfloor$. $f(C)$ refers to the configuration fitness function value, (i.e. the dependability measure given by equation (2).) All methods use the same problem space, each answer is a J-dimensional vector. Theorem 1 is used to trivially reject configurations not within region (2) for genetic algorithms and simulated annealing but not for tabu search.. A brief discussion of each heuristic is contained in this section.

4.1 Genetic Algorithms

Genetic algorithms apply the concept of survival of the fittest to optimization problems. Possible solutions to a problem are called *chromosomes* and a diverse set of chromosomes are grouped into a *gene pool*. The relative quality of these answers are determined using a *fitness function* $f(C)$. This determines whether or not a chromosome will be used in producing the next generation. The next generation is generally formed via the processes of *crossover*: combining elements of two chromosomes from the gene pool, and *mutation*: randomly altering elements of a chromosome. Refer to [15] for details. Several reproduction strategies exist in the literature. We apply an *elitist strategy*, given in [4], to the configuration problem.

Algorithm: genetic_search
Inputs: J, d_i for $1 \le i \le J$, c_i for $1 \le i \le J$, and D.
Outputs: Vector L, the most dependable configuration.
Procedure:
Step 1: Generate initial gene pool *GP* of 150 integer vectors of length J. The size, 150, of *GP* was determined by experimentally. *GP* contains all solo configurations of type (0,...0,N_i,0...0) and random combinations of components. If an element of *GP* is not within region (2) a component is added or removed at random until it is.
Step 2: for k = 1 to 500 begin
  /* percents for elite, crossover and mutation and the number of iterations found experimentally */
  for h = 1 to 150 begin
    fitness[h] = f(*GP*[h])
  end
  for h = 1 to 30 begin /* keep the elite */
    GP_next[h] = the h'th most dependable configuration in *GP*.
  end
  for h = 31 to 142 begin /* crossover*/
    GP_next[h] = randomly combine 2 random configurations from *GP*.
  end
  for h = 143 to 150 begin /* mutants */
    GP_next[h] = random configurations.
  end
  for h = 1 to 150 begin
    while(*GP_next*[h] is not in region (2)) add or subtract a random component
  end
  GP = *GP_next*
 end
Step 5: for h = 1 to 150 begin
    fitness[h] = dependability of configuration *GP*[h]
 end
L = the most dependable configuration in *GP*

Figure 4 Pseudo-code for genetic algorithm.

The solution space to this problem consists of component configurations. The chromosomes used by the genetic algorithm consist of a vector which describes a possible system configuration. Position i of the vector (where i is between 1 and J) is an integer ranging from 0 to N_i giving the number of components of type i in the system. The elitist reproduction strategy used consists of three major steps. First, the best 20% of the gene pool is copied intact into the gene pool for the next generation. Second, 75% of the next generation is determined by randomly mixing elements from two chromosomes chosen at random from the gene pool of the current generation. The remaining 5% of the next generation consists of random mutations. All resulting chromosomes are checked to be sure that they are within region (2). If not, a random component is added or removed until the chromosome is. This strategy is stable since the quality of the best answers must be monotonically increasing. Mutations are useful since they provide new input to the algorithm. This guards against convergence to a sub-optimal answer.

We initialize the gene pool with a set of reasonable answers. They include all single component type configurations, and many random configurations. The reproduction scheme determines the following generations. The algorithm is performed for 500 generations and the best solution present in the gene pool at that point is taken to be the configuration proposed by the genetic algorithm. There are no deterministic means of finding the values for many parameters of a genetic algorithm: such as gene pool size, crossover rate, mutation rate, and stopping criteria, these values have been determined experimentally. The ones we used are given in the pseudo-code. We used a genetic algorithm as proposed by Holland [15] and modified it to suit our application. The algorithm is summarized in figure 4.

4.2 Simulated Annealing

Simulated annealing attempts to find optimal answers to a problem in a manner analogous to the formation of crystals

in cooling solids. A material heated beyond a certain point will become fluid, if the fluid is cooled slowly the material will form crystals and revert to a minimal energy state. For a broader treatment of this topic refer to [19].

```
Algorithm: simulated_annealing
Inputs: J, d_i for 1 ≤ i ≤ J, c_i for 1 ≤ i ≤ J, and D.
Outputs: Vector L with most dependable configuration.
Procedure:
Step 1: Compute dependability of all containing only 1
    component type
Step 2: Make the component with the highest
    dependability type 1.
Step 3: CC = (N_1, 0, ....,0) /* starting point */
  τ = 1.0              /* Initial temperature */
Step 4: CC_mod = 1                  step4_iter = 0
  While (CC_mod != 0) and
       (step4_iter < maximum number for step 4) do
  begin
    CC_mod = 0                  inner_loop_iter = 0
    While(CC_mod < maximum transitions) and
      (inner_loop_iter < maximum inner loop) do
        begin
          new_CC = random modification of CC
          while(new_CC is not in region (2))
              add or subtract a random component.
          ΔC = f(new_CC) - f(CC)
          if(ΔC > 0) then
            begin
              CC = new_CC
              CC_mod = CC_mod + 1
            end
        else with probability of Boltzmann distribution
              ΔC and τ do
          begin
              CC = new_CC
              CC_mod = CC_mod + 1
          end
        inner_loop_iter = inner_loop_iter + 1
      end
    τ = 0.9 * τ
    step4_iter = step4_iter + 1
  end
Step 5: Output CC as the most dependable configuration
.
```

Figure 5 Pseudo-code for simulated annealing

The strategy of the algorithm uses the *fitness function* $f(C)$ to compare the relative merit of various points in the problem space. As before, the problem space is described by vectors corresponding to possible system configurations, and the fitness function is equation (2). The algorithm starts at the most dependable configuration made of one component. From the algorithm's current position a neighboring point is chosen at random. The cost difference between the new point and the current point is calculated. This difference is used together with the current system temperature to calculate the probability of the new position being accepted. This probability is given by a Boltzmann distribution $e^{-\Delta C/\tau}$. The process continues with the same temperature τ for either a given number of iterations, or until a given number of positions have been occupied, at which time τ is decreased. The temperature decreases until no transitions are possible, so the system remains frozen in one position. This occurs only when ΔC is positive for all neighboring points, therefore the position must be a local minimum and may be the global minimum[23]. To maximize the fitness function it is necessary to set the sign of ΔC appropriately.

The simulated annealing method used in our research is based on the algorithm given in [19,23]. The algorithm has been modified so that the parameters being optimized and the fitness function are appropriate for our application, and a cooling schedule has been found which allows the algorithm to converge to a reasonable solution.

4.3 Tabu Search

Tabu search modifies an existing heuristic search by keeping a list of the nodes in the search space visited most recently by the search. These points become "tabu" meaning that they are not revisited while on the list. This allows a search algorithm to descend from local maxima. Our implementation uses a list that is infinite. A detailed explanation of tabu search can be found in [14,16].

Figure 5 illustrates a greedy heuristic starting at A. It remains at A since A has no neighbors with a larger value. Point A is a local maxima and a greedy heuristic would return A (5.0) as the maximum. This is incorrect as the maximum value is at D (6.1). Tabu search puts points visited on a list and forbids movement to these points. At A the tabu list is {A} and the only neighbor not in the tabu list is B. From B the only neighbor not on the list is C. From C tabu search moves directly to the global maximum D.

The tabu search used here relies on a "greedy" heuristic and starts with the most dependable configuration of only one component type. The search adds or subtracts one to each x_i. Configurations on the tabu list, configurations with negative numbers of components, and configurations which exceed the cost constraint are disqualified. The algorithm evaluates each configuration using the fitness function. The search moves to the configuration with the maximum value for configuration dependability. When a configuration is visited it is placed on the tabu list, should the search return to its neighborhood later, the fitness function is set to zero. As each configuration is visited, the value of configuration dependability is compared to the largest value up to that point. If the value is

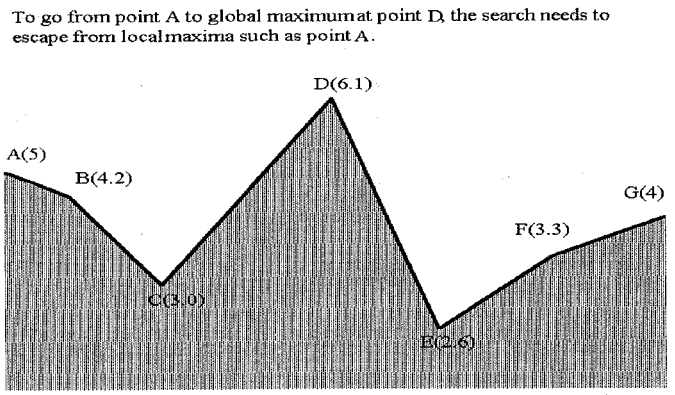

Figure 5 One dimensional search space with local maxima.

larger, the configuration becomes the best fit found. Figure 6 presents pseudo-code for this implementation of tabu search.

Since no clear stopping criteria exists for tabu search, this study was done by comparing the results from a given number of iterations of the tabu search and genetic algorithms with results obtained by performing simulated annealing which ran until completion.

5 Experimental Results

Tables 1, 2 and 3 present results from an example problem. The example shown consists of 11 possible component types which can be used to construct a sensor system. For each component type the per item cost, failure

Algorithm: tabu_search
Inputs: J, d_i for $1 \le i \le J$, c_i for $1 \le i \le J$, and D.
Outputs: Vector L with the most dependable configuration.
Procedure:
Step 1: Compute $f(C)$ of all solo configurations. The best solo configuration type becomes type 1.
Step 2: $x_1 = N_1$ $x_i = 0$, $2 \le i \le J$
$Best\_Conf = (x_1, x_2, \ldots, x_J)$ $Current\_Conf = (x_1, x_2, \ldots, x_J)$
For $I := 1$ to N do /* N is determined experimentally */
Step 3: Compute dependability for all $2J$ neighbors of *Current_Conf*. A neighbor is a configuration which is made by adding or subtracting 1 component from any x_i ($2 \le i \le J$). A negative number of components is not allowed. Configurations which exceed cost bounds are disqualified. If any neighbor is on the tabu list set the dependability for that neighbor to zero.
Step 4: Set *Current_Conf* to the most dependable neighbor.
Step 5. If $f(Current\_Conf) > f(Best\_Conf)$ then $Best\_Conf = Current\_Conf$
Step 6. Append *Current_Conf* to tabu list.
end for loop.
Step 7. return(*Best_Conf*)

Figure 5 Pseudo-code for tabu search.

Table 1

Sensor	Cost	Failure	Repair	Solo	Dep. 52	Solo	Dep. 58
1	$20.00	0.06	0.3	2	69%	2	69%
2	$10.00	0.15	0.3	5	79%	5	79%
3	$20.00	0.13	0.81	2	74%	2	74%
4	$5.00	0.5	0.95	10	76%	11	86%
5	$25.00	0.11	0.9	2	79%	2	79%
6	$15.00	0.32	0.84	3	81%	3	81%
7	$7.00	0.07	0.1	7	69%	8	57%
8	$8.00	0.22	0.59	6	79%	7	91%
9	$20.70	0.01	0.07	2	77%	2	77%
10	$6.80	0.19	0.35	7	80%	8	70%
11	$7.00	0.22	0.4	7	79%	8	70%

rate and repair rate are given, as is the cost constraint which must be respected. Table 1 shows the salient characteristics of each sensor component. Table 2 gives the sensor configurations found by all three heuristics when two separate cost limits ($52.00 and $58.00) are used. This illustrates how changing the cost constraint modifies the system. This problem defines a non-linear problem space of 11 dimensions. For this and higher dimension problems, any exhaustive search method would be impractical. It is therefore important to find heuristic methods capable of providing near-optimal results.

Note that in table 3 the heuristics provide results which differ from the others. These answers are significantly superior to the dependability of configurations made up of a single component type. We call these configurations *solo configurations*. Solo configurations represent current practice where heterogeneous redundancy may not be considered. The configurations in table 2 are, with one exception, significantly more reliable than those currently used.

When the cost constraint is $52.00 the system configuration found by both simulated annealing and genetic algorithms is 14% more dependable than the most dependable solo configuration. Note that changing the cost constraint to $58.00 radically changes the answers found by all three methods. This points out the volatility of the problem space and the difficulty of finding optimal configurations. Increasing the number of components in some of the solo configurations is shown to decrease system dependability. This is due to the nonlinear nature of the problem space which is extremely jagged .

As implemented, the heuristics always find an answer which is at least as good as the most dependable solo configuration. That solo configuration is used as an initial condition for the heuristics. If significantly better configurations exist, the heuristics tend to identify configurations which are superior to the solo configuration. Due to the non-deterministic nature of simulated annealing and genetic algorithms and the non-linearity of the problem no theoretical performance guarantee is possible. Our results

show that, in spite of this, the practical results are very promising.

Table 2

	Cost Limit 52.00			Cost Limit 58.00		
Sensor	Tabu Max	SA Max	GA Max	Tabu Max	SA Max	GA Max
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	1	0	0	0	0	0
5	0	0	0	0	2	1
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	3	1	1	7	1	1
9	1	2	2	0	0	1
10	0	0	0	0	0	0
11	0	0	0	0	0	0

Table 3

	Components	Cost	Depend
Cost limit	$52.00		
Tabu	5	$49.70	89%
SA	3	$49.40	92%
GA	3	$49.40	92%
Cost limit	$58.00		
Tabu	7	$56.00	91%
SA	3	$58.00	94%
GA	3	$53.70	93%

6 Discussion

The results given in section 5 illustrate several important points. Heuristics provided configurations which significantly increased system dependability relative to the solo configurations. None of the approaches used is guaranteed to find the optimal configuration. That could only be guaranteed by an exhaustive search of the entire search subspace defined by theorem 1. For problems with J component types this search space has J dimensions and an exhaustive search becomes computationally intractable for only moderately sized values of J.

Simulated annealing consistently found solutions which were as good as, or better, than the solutions found by the other two methods. The results found by tabu search were often inferior to the results of both genetic algorithms and simulated annealing. Since the quality of the results found by each heuristic depends on implementation details, such as the cooling schedule for simulated annealing, the reproduction strategy for genetic algorithms and the length of the tabu list for tabu search, it is possible that modifications of these parameters could result in programs which provide answers superior to those found by our implementations.

The use of at least two heuristics for this problem is recommended to provide an independent verification of the quality of the configuration chosen. Our results indicate that simulated annealing and genetic algorithms are the most promising heuristics for this application.

This paper presents a methodology for sensor selection which successfully produces dependable multi-sensor configurations from components whose dependability statistics are known. These results are significant because:

- Near optimal dependability is maintained within cost..
- Cost may be either a dollar sum or system weight.
- Computational results demonstrate the method.
- Implementation details are provided.
- They find good configurations for multi-sensor system.

The resulting systems are more dependable than current practice. While this methodology is of general interest, it is especially significant for space and aeronautics applications where weight and dependability are factors of importance.

Acknowledgment

This work was supported in part by the Office of Naval Research grant N 00014-94-1-0343.

References

[1] J. G. Balchen and F. Dessen "Structural Solution of Highly Redundant Sensing in Robotics Systems" in *Highly Redundant Sensing in Robotics Systems* (ed.s Tou and Balchen) NATO Advanced Science Institutes Series vol. F-58, Springer Verlag, pp. 263-275, 1990.

[2] M. Barborak, M. Malek, and A. Dahbura, "The Consensus Problem in Fault Tolerant Computing", *ACM Computing Surveys* ,25, 2(June), pp. 171-220, 1993.

[3] R. Barlow and F. Proschan, *Mathematical Theory of Reliability*, Wiley, New York, 1965.

[4] J. C. Bean, "Genetic Algorithms and Random Keys for Sequencing and Optimization", *ORSA Journal on Computing*, 6, 2(Spring), pp. 154-160, 1994.

[5] R. R. Brooks, N. S. V. Rao and S. S. Iyengar, "Resolution of Contradictory Sensor Data", *Journal of Intelligent Automation and Soft Computing*, accepted for publication, 1995.

[6] R. R. Brooks and S. S. Iyengar, "Algorithm for Resolving Inter-Dimensional Inconsistencies in Redundant Sensor Arrays", in *Parallel and Distributed Signal and Image Integration Problems*, Proceedings of Indo-US Workshop December 1993 Pune, India, World Scientific, Singapore, pp. 254-262, 1995

[7] R. R. Brooks and S. S. Iyengar, "Self-Calibration of A Noisy Multiple Sensor System with Genetic Algorithms",

Self-Calibrated Intelligent Optical Sensors and Systems, SPIE, Bellingham, Wa., Proceedings of SPIE International Symposium on Intelligent Systems and Advanced Manufacturing, Philadelphia, Pa. October 1995

[8] R. R. Brooks and S. S. Iyengar, "Methods of Approximate Agreement for Multisensor Fusion", *SPIE Proceedings Signal Processing, Sensor Fusion and Target Recognition IV*, Orlando, Fla. April 1995.

[9] R. R. Brooks, S. S. Iyengar, and J. Chen, "Automatic Correlation and Calibration of Noisy Sensor Readings using Elite Genetic Algorithms", Artificial Intelligence, accepted for publication 1996.

[10] R. R. Brooks and S. S. Iyengar, "Approximate Agreement for Distributed Computing by Masking Data Conflicts", IEEE Computer, accepted for publication to appear in 1996.

[11] R. R. Brooks *Robust Sensor Fusion Algorithms: Calibration and Cost Minimization*, Doctoral Dissertation, Louisiana State University, 1996.

[12] K. L. Chung, *Elementary Probability Theory with Stochastic Processes*, Springer Verlag, Heidelberg, 1974.

[13] R. Faure, *Guide de la Recherche Opérationelle, tome 1 les fondements*, pp. 159-164, Editions Masson, Paris, 1986.

[14] F. Glover, "Tabu Thresholding: Improved Search by Nonmonotonic Techniques", *ORSA Journal on Computing*, vol. 7, No. 4(Fall), pp. 426-442, 1995.

[15] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[16] T. C. Hu, A. B. Kahng and C. A. Tsao, "Old Bachelor Acceptance: A New Class of Non-Monotone Threshold Accepting Methods", *ORSA Journal on Computing*, vol. 7, No. 4(Fall), pp. 417-425, 1995.

[17] S. S. Iyengar, L. Prasad, and H. Min, *Advances in Distributed Sensor Integration: Applications and Theory*, Prentice Hall, 1995.

[18] J. Kershaw "Dependable Systems Using VIPER" in *Fault Tolerant Design Concepts for Highly Integrated Flight Critical Guidance and Control Systems* NATO Advisory Group for Aerospace Research and Development Conference Proceedings No. 456 AGARD-CP-456, pp. 25-1-25-7, 1990.

[19] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, D. Reidel Publishing Co. , Dordrecht, 1987.

[20] L. Lamport, R. Shostak, and M. Pease "The Byzantine Generals Problem", *ACM Trans. Program. Lang. Syst.*, 4, 3(July), pp. 382-401, 1982.

[21] R. Luo and M. Kay "Data Fusion and Sensor Integration: State-of-the-art 1990s" *Data Fusion in Robotics and Machine Intelligence* Abidi and Gonzales ed.s, Academic Press, Boston, pp. 7-136, 1992.

[22] J. L. Michaloski, P.G. Backes, and R. Lumin "Integration of Sensor Feedback and Teleoperation into an Open Standard" in *Sensor Fusion and Networked Robotics VIII* (eds. Schenker and McKee), SPIE Proceedings vol. 2589, pp. 206-217, October, 1995.

[23] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in Fortran*, pp. 436-448, Cambridge University Press, 1986.

[24] T. Sadeghi and G. Mayville "Fault-tolerant Flight-Critical Control Systems" in *Fault Tolerant Design Concepts for Highly Integrated Flight Critical Guidance and Control Systems* NATO Advisory Group for Aerospace Research and Development Conference Proceedings No. 456 AGARD-CP-456, pp. 26-1-26-12, 1990.

[25] D. P. Siewiorek and R. S. Swarz, *Reliable System Design and Evaluation*, Digital Press, Bedford, Mass, 1992.

[26] G. Strang, *Linear Algebra and Its Applications*, Academic Press, New York, 1976.

[27] J. T. Tou "A Knowledge-Based System for Redundant and Multi-Sensing in Intelligent Robots" in *Highly Redundant Sensing in Robotics Systems* (ed.s Tou and Balchen) NATO Advanced Science Institutes Series vol. F-58, Springer Verlag, pp.3-14, 1990.

[28] J. Wahrburg "Control Concepts for Industrial Robots Equipped with Multiple and Redundant Sensors" in *Highly Redundant Sensing in Robotics Systems* (ed.s Tou and Balchen) NATO Advanced Science Institutes Series vol. F-58, Springer Verlag, pp.277-291, 1990.