# Efficiently Solving the Redundancy Allocation Problem Using Tabu Search

Sadan Kulturel-Konak[1], Alice E. Smith[1], David W. Coit[2]

[1]Department of Industrial and Systems Engineering
Auburn University
207 Dunstan Hall
Auburn, Alabama 36849
334-844-1400
aesmith@eng.auburn.edu

[2]Department of Industrial Engineering
Rutgers University
96 Frelinghuysen Rd.
Piscataway, New Jersey 08554
732-445-2033
coit@rci.rutgers.edu

**Abstract:** A tabu search meta-heuristic has been developed and successfully demonstrated to provide solutions to the system reliability optimization problem of redundancy allocation. Tabu search is particularly well-suited to this problem and it offers distinct advantages compared to alternative optimization methods. While there are many forms of the problem, the redundancy allocation problem generally involves the selection of components and redundancy levels to maximize system reliability given various system-level constraints. This is a common and extensively studied problem involving system design, reliability engineering and operations research. It is becoming increasingly important to develop efficient solutions to this reliability optimization problem because many telecommunications (and other) systems are becoming more complex, yet with short development schedules and very stringent reliability requirements. Tabu search can be applied to a more diverse problem domain compared to mathematical programming methods, yet offers the potential of greater efficiency compared to population-based search methodologies, such as genetic algorithms. The tabu search is demonstrated on numerous variations of three different problems and compared to integer programming and genetic algorithm solutions. The results demonstrate the benefits of tabu search for solving this type of problem.

Accepted to *IIE Transactions*

September 2002

# Efficiently Solving the Redundancy Allocation Problem Using Tabu Search

Sadan Kulturel-Konak[1], Alice E. Smith[1], David W. Coit[2]

[1]Department of Industrial and Systems Engineering
Auburn University
207 Dunstan Hall
Auburn, Alabama 36849
334-844-1400
aesmith@eng.auburn.edu

[2]Department of Industrial and Systems Engineering
Rutgers University
96 Frelinghuysen Rd.
Piscataway, New Jersey 08554
732-445-2033
coit@rci.rutgers.edu

## 1. INTRODUCTION

A well-known and complex reliability design problem is the redundancy allocation problem (RAP). Realistic formulations of the RAP are characterized by a large combinatorial search space with multiple constraints. Generally, either system reliability is maximized or system cost is minimized given system level constraints on reliability, cost, weight, power, etc. The problem has previously been solved using many different optimization approaches and for different problem formulations [1]. Tabu search (TS) has several potential advantages for solving this problem yet it has not previously been demonstrated or extensively tested to evaluate its effectiveness in this particular setting. An efficient TS is described and demonstrated here, named TSRAP, and the results are compared with other published approaches to the problem.

The RAP is useful for system designs that are largely assembled and manufactured using off-the-shelf components, and also, have very high reliability requirements. Most electronic systems are in this category. Redundancy allocation involves the selection of components from among discrete choices to perform defined functions. For each required component, there are multiple, or even numerous, possible selections available from different component vendors, with different costs, manufacturing, testing and quality assurance provisions. Additionally, for systems to achieve the required reliability levels, there is often the need to implement redundancy where multiple components are available to continue to perform the required functions in the event of a component failure. Solutions to the RAP intend to identify the optimal combination of component selections and redundancy levels given constraints on the overall system.

Mathematical programming techniques, such as dynamic programming and integer programming (IP) have been successfully applied to variations of the problem. Often to apply these methods, it has been necessary to artificially restrict the search space to solutions where only one component type can be selected for each subsystem, and then the same type can be used to provide redundancy. Once this restriction has been imposed, transformations can be applied to the objective function, and then, mathematical programming used to obtain the optimal solution. Unfortunately, these restrictions are necessary for application of the optimization strategies, but not for the actual engineering design problem. In practice, different components, performing the same function, can be used within a system to provide high reliability. For example, many airplanes are designed with both an electronic and mechanical gyroscope. They perform the same function but they have other characteristics that are different. Thus, mathematical programming approaches to the problem yield "optimal solutions," but for an artificially restricted search space. In practice, it is possible to obtain solutions that are superior by relaxing the restriction, as noted by Coit & Smith [2].

Genetic algorithms (GAs) have been able to overcome some the deficiencies of mathematical programming approaches, and for many problems, the use of GAs has provided excellent results. However, GA is a population-based search requiring the evaluation of multiple prospective solutions (i.e., a population) over many generations. Thus, for some complex problems, this results in significant computational effort and a more efficient approach to the problem is desirable if it can yield comparable, or even better, results.

TS is a competing meta-heuristic method for many of the same large and complex combinatorial optimization problems that have been optimized by GA. It is a conceptually simple solution technique that moves deterministically through successive iterations by considering neighboring moves. (Probabilistic versions of TS also exist.) Although its origins go back to the late 1960s and early 1970s, TS was proposed in its current form in the late 1980s by Glover [3]. Like other meta-heuristics, it is difficult to specify a "canonical form." However, most TS versions can be characterized by the following two important properties: *i*) complementing local search (the neighborhood concept), and *ii*) prohibiting moves that have been previously selected (the adaptive memory concept). General material and additional background information on TS is available from several references [4 - 8], the most comprehensive being Glover & Laguna [7].

Beginning with an initial feasible solution, successive "moves" to superior solutions are made within a neighborhood. To avoid convergence to a local optimum, particular moves (most often those recently taken) are temporarily deemed to be "tabu" or forbidden, allowing for a more diverse search. Implementation of TS requires problem-specific definition of a neighborhood, memory through use of a tabu list, and aspiration criteria that may override the tabu list (e.g., if the solution is the best yet identified). Like GAs, TS does not require objective function gradient information. This is particularly important because the objective function for some forms of the RAP, such as the multiple *k*-out-of-*n* problem, is not differentiable. Unlike

GAs, TS is not population-based, and successively moves from solution to solution. This offers some potential for improved efficiency if it also provides the same quality of solutions. (It should be noted that a population-based version has been effectively implemented for TS by means of the scatter search and path relinking strategies, whose applications are surveyed by Glover & Laguna [9].)

Given the natural neighborhood structure of the RAP, it seems particularly amenable to TS. If the neighborhood is defined within a single subsystem, as done in this paper, when the search moves to a new solution, then only the design for one subsystem is affected, and the objective function can be updated efficiently without re-calculation of the unaffected subsystems. However, because of the global changes effected through the crossover and mutation of GA, each newly produced solution within a GA requires re-calculation of the system reliability.

## 1.1 Notation

| | |
|---|---|
| $R(t,\mathbf{x})$ | system reliability at time $t$, depending on $\mathbf{x}$ |
| $\mathbf{x}$ | $(x_{11}, x_{12}, ..., x_{1,m_1}, x_{21}, x_{22}, ..., x_{2,m_2}, x_{31}, ..., x_{s,m_s})^T$ |
| $x_{ij}$ | quantity of the $j^{th}$ available component used in subsystem $i$ |
| $m_i$ | number of available components for subsystem $i$ |
| $s$ | number of subsystems |
| $\mathbf{n}$ | $(n_1, n_2, ..., n_s)$ |
| $n_i$ | total number of components used in subsystem $i$, $n_i = \sum_{j=1}^{m_i} x_{ij}$ |
| $n_{\max,i}$ | upper bound for $n_i$ ($n_i \le n_{\max,i} \ \forall\ i$) |
| $C(\mathbf{x})$ | system cost depending on $\mathbf{x}$ |
| $W(\mathbf{x})$ | system weight depending on $\mathbf{x}$ |
| $C, W, R$ | system-level constraint limits for cost, weight, and reliability |
| $c_{ij}, w_{ij}, r_{ij}$ | cost, weight, and reliability for the $j^{th}$ available component for subsystem $i$ |
| $t_o$ | mission time (fixed) |
| $\mathbf{k}$ | $(k_1, k_2, ..., k_s)$ |
| $k_i$ | minimum number of operating components required for subsystem $i$ |

| | |
|---|---|
| $\lambda_{ij}$ | parameter for exponential distribution, $f_{ij}(t) = \lambda_{ij} \exp(-\lambda_{ij} t)$ |
| $\mathrm{NFT}_{i,j}$ | Near Feasible Threshold for the $i^{\text{th}}$ constraint at the $j^{\text{th}}$ move of the TS |
| $F_j$ | number of feasible solutions on the tabu list |
| $T_j$ | total number of solutions on the tabu list |
| $\rho_j$ | feasibility ratio, $\rho_j = F_j/T_j$ |

## 2. REDUNDANCY ALLOCATION PROBLEM

The most studied design configuration of the RAP is a series system of $s$ independent $k$-out-of-$n$:G subsystems (Figure 1). A subsystem is functioning properly if at least $k_i$ of its $n_i$ components are operational. If $k_i$ is one for all subsystems, then it is a series-parallel system. The RAP is NP-hard [10] and has been studied in many different forms. An overview and summary of work for different approaches to RAP is presented in Tillman, Hwang & Kuo [11], and more recently by Kuo & Prasad [1].

The RAP can be formulated with system reliability as the objective function (more typically found in the literature) or in the constraint set (often found in actual engineering design problems). Problem P1 maximizes system reliability given overall restrictions on system cost of $C$ and system weight of $W$. Problem P2 minimizes system cost given overall restrictions on maximum system weight of $W$ and minimum system reliability $R$. For these formulations, it is assumed that system weight and system cost are linear combinations of component weight and cost. However, this is not a restriction for the TS methodology. It can easily accommodate more constraints and more complex functions, as well.

Problem P1: reliability maximization

max $\quad R(t_o;\mathbf{x})$

s.t. $\quad \sum_{i=1}^{s}\sum_{j=1}^{m_i} c_{ij} x_{ij} \le C$

$$\sum_{i=1}^{s}\sum_{j=1}^{m_i} w_{ij}x_{ij} \le W$$

$$\sum_{j=1}^{m_i} x_{ij} \ge k_i \text{ for } i = 1, \ldots, s$$

$$x_{ij} \in \{0, 1, 2, \ldots\}$$

<u>Problem P2:</u> cost minimization

$$\min \quad C(\mathbf{x}) = \sum_{i=1}^{s}\sum_{j=1}^{m_i} c_{ij}x_{ij}$$

$$\text{s.t.} \quad R(t_o;\mathbf{x}) \ge R$$

$$\sum_{i=1}^{s}\sum_{j=1}^{m_i} w_{ij}x_{ij} \le W$$

$$\sum_{j=1}^{m_i} x_{ij} \ge k_i \text{ for } i = 1, \ldots, s$$

$$x_{ij} \in \{0, 1, 2, \ldots\}$$

The series-parallel RAP (i.e., $k_i = 1 \;\forall\; i$) has been widely studied. The different approaches including dynamic programming (Bellman [12], Bellman & Dreyfus [13, 14], Fyffe, Hines & Lee [15], Nakagawa & Miyazaki [16]), IP (Ghare & Taylor [17], Bulfin & Liu [18], Misra & Sharma [19], Gen et al. [20]), and mixed-integer and nonlinear programming (Tillman, Hwang & Kuo [21]). More recently, meta-heuristic methods such as GA (Painton & Campbell [22], Coit & Smith [2, 23], Coit et al. [24]), TS (Kulturel-Konak et al. [25]), and the Ant System Algorithm (Liang & Smith [26]) have been applied to the problem.

An IP approach was used by Coit & Liu [27] to solve the RAP with *k*-out-of-*n* subsystems and exponential component failure times. For this formulation, it was necessary to assume that only one component type (of the $m_i$ options) is allowed in a particular subsystem. The problem was transformed to the form of a standard 0-1 IP problem with $\sum_{i=1}^{s}(n_{\max,i} - k_i + 1)m_i$

decision variables.  Optimal solutions can be found using standard algorithms developed specifically for 0-1 IP.  In this paper, TS is applied to this same problem class.  It is no longer necessary to restrict the solution space when using TS, and better solutions (i.e., higher system reliability) can be identified.

For the restricted search space (no component mixing), reliability for a $k$-out-of-$n$ subsystem with active redundancy is calculated as the sum of ($n_i$-$k_i$+1) binomial probability mass functions.  The system reliability is the product of the subsystem reliability values, and is given by:

$$R(t;\mathbf{x}\in S_r)=\prod_{i=1}^{s}\sum_{\substack{l=k_i\\(x_{ij}\geq k_i)}}^{x_{ij}}\binom{x_{ij}}{l}\left(\exp(-\lambda_{ij}t)\right)^{l}\left(1-\exp(-\lambda_{ij}t)\right)^{x_{ij}-l} \tag{1}$$

$S_r$ is a restricted solution set.  $S_r$ is restricted to solutions such that for all $i$, there exists a unique $p$ ($p\in\{1, 2, \ldots, m_i\}$) with $x_{ip} \geq k_i$ and $x_{ij} = 0\ \forall\ j \neq p$.  The set $S_r$ requires that only one component type is chosen for each subsystem.

The RAP has been solved using many different approaches.  One reason why this problem has been so extensively studied is because there does not seem to be a solution methodology that is universally superior.  TS offers another alternative that provides greater flexibility than mathematical programming, but potentially greater efficiency than GA. According to Kuo & Prasad [1], "a well designed TS can offer excellent solutions in large system reliability optimization."  Despite this recommendation of the promise of TS, its effectiveness in solving the RAP has not been sufficiently documented, nor demonstrated.

While TS has not been extensively tested on the RAP, it has been applied successively to related problems.  TS has been demonstrated on structural design problems with reliability constraints by Bland [28, 29], and telecommunications network design problems with unreliable components by Xu et al [30] and Pierre & Elgibaoui [31].  Brooks et al. [32] demonstrated the use of TS to determine the optimal configuration of distributed sensors based on the sensor reliability. Despite the demonstrated usefulness of TS, a primary deficiency, also found in other meta-heuristics, is that there is no guarantee that the global optimal solution will be identified.

## 3. TABU SEARCH APPROACH

The RAP has an inherent neighborhood structure that engenders single solution meta-heuristics such as simulated annealing (SA) and TS. While many papers have reported good results with SA, there is also common agreement that the method can be quite sensitive to annealing schedule. SA makes stochastic moves using an acceptance probability. However the moves of TS are deterministic, reducing variability to both initial solution and to other search parameters. These considerations motivated the selection of TS as a promising meta-heuristic for solving the RAP. The TS presented here, TSRAP, is based on the implementation developed by Kulturel-Konak et al. [25].

### 3.1 Tabu Search Meta-Heuristic for the RAP

For the series parallel version of the RAP, the encoding is a permutation encoding of size $\sum_{i=1}^{s} n_{\max,i}$ representing a concatenation of the components in each subsystem including non-used components (i.e., defined as “blanks” when $n_i < n_{\max,i}$).

TS involves the determination of a tabu list of unavailable moves. For TSRAP, the structure (encoding) of the subsystem that has been changed in the accepted move is added to the tabu list. The content of the tabu list is very influential on the performance of TS. In this case, a more specific entry (such as the structure of all subsystems for an accepted move) would not be limiting enough. An insufficient number of moves would be tabu, having the potential effect of stalling the search in a local optimum. A less specific entry (such as the components altered in the accepted move) would constrain the non-tabu moves available too greatly. This may have the effect of a coarse search; that is, one that gets near the global optimum, but cannot arrive at the exact optimal configuration due to move restrictions. Since moves operate on subsystems, it was natural to also use a subsystem based tabu entry. A dynamic length tabu list is used as it usually found that this reduces sensitivity of the algorithm to selection of the tabu list length. The tabu list length is re-set every 20 iterations to an integer value distributed uniformly between [$s$, $3s$] and [$14s$, $18s$] for Problems P1 ($s$=14) and P2 ($s$=2), respectively. This resulted in tabu list sizes from 14 to 42 in the first instance and from 28 to 36 in the second instance. The algorithm is not sensitive to list size and a correlation with $s$ is not necessary. An aspiration criterion is also required to determine when a particular move on the tabu list becomes available again.

The following terms are used below: BEST MOVE (this is the best solution that would result from taking any of the currently available moves), BEST SO FAR (this is the best solution found so far in the search – it may be feasible or infeasible), and BEST FEASIBLE SO FAR (this is the best feasible solution found so far in the search). TSRAP proceeds as follows:

***Step 0*** Generate a feasible random initial solution

To obtain an initial feasible solution, $s$ integers between $k_i+1$ and $n_{max,i}–2$ (inclusive) are chosen according to a discrete uniform distribution to represent the number of components in parallel ($n_i$) for each subsystem. The algorithm is not sensitive to starting solution; this procedure typically generates a solution with an average number of components per subsystem. Then, the $n_i$ components are assigned according to a discrete uniform distribution to the $m_i$ different types. If feasible, it becomes the initial solution. If not, then the process is repeated until a feasible solution is found.

***Step 1*** <u>Search the neighborhood of all possible defined moves for each subsystem</u>[1]

Moves operate on subsystems only and two kinds are used.

a) For the TSRAP that allows component mixing within a subsystem, the first type of move is to change the number (i.e., quantity) of a particular component type by adding or subtracting one ($x_{ij} \rightarrow x_{ij}+1$, or $x_{ij} \rightarrow x_{ij} - 1$). These moves are considered individually for all available component types within all subsystems. The second type of move is to simultaneously add one component to a certain subsystem and delete another component, of a different type, within the same subsystem ($x_{ij} \rightarrow x_{ij}+1$, and $x_{ik} \rightarrow x_{ik} - 1$ for $j \neq k$, enumerating all possibilities).

b) For the TSRAP without allowing component mixing, the first type of move is to change the number of components by adding or subtracting one ($x_{ij} \rightarrow x_{ij}+1$, or $x_{ij} \rightarrow x_{ij} - 1$), for all subsystems. These moves are considered individually for all available component types within all subsystems. The second type of move is to change the type of component choice ($x_{ij} \rightarrow x_{ik}$ for $j \neq k$), for each subsystem, by trying all available component choices.

An important advantage of these types of moves is that they do not require recalculating the entire system reliability. Subsystems are changed one-at-a-time. Therefore, only the reliability of the changed subsystem is recalculated and system reliability is updated accordingly. The two types of moves are performed independently on the current solution, and the best among them is selected.[2] If this solution, the BEST MOVE, is tabu and the corresponding solution is not better than the BEST SO FAR solution (i.e., the aspiration

[1] For larger problems, it will be necessary to limit the neighborhood other than by the move definitions, as done here. An effective method to accomplish this is through use of a candidate list [7].

[2] Note that the selected move is not necessarily better than the current solution – it is simply the best available move.

criterion is not satisfied), then the move is disallowed, and *Step 1* is repeated. If the solution is not tabu or if it is tabu, but also better than the BEST SO FAR solution, then it is accepted.

***Step 2*** Update the tabu list

The structure of the subsystem that has been changed in the accepted move is added to the tabu list. If the tabu list is full, the oldest tabu list entry is deleted. To know if an entry on the tabu list is feasible or infeasible, the system cost and weight are stored with the subsystem structure involved in the move within the tabu list.

***Step 3*** Check stopping criterion

Next, the stopping criterion is checked. It is defined as the maximum number of iterations without finding an improvement in the BEST FEASIBLE SO FAR. If it is reached, the search is concluded and the BEST FEASIBLE SO FAR solution is the TSRAP recommended solution. Otherwise, return to *Step 1*.

## 3.2 Penalty Function

The search intends to find the best feasible solution. To maintain feasibility, all infeasible solutions could be rejected when encountered or they could be penalized and allowed within the search. For solving the RAP with a GA search, Coit & Smith [23] observed that better final feasible solutions could be found by allowing the search to proceed through the infeasible region, but by penalizing those solutions based on the degree of infeasibility. The best results were observed when the search was intensified near the feasible/infeasible boundary, considering prospective solutions on both sides of the boundary. The idea of exploring around boundaries is an old one in TS, and is refined in the TS notion of strategic oscillation. (See, for example, Section 4.4 of Glover & Laguna [7].) Based on numerous findings of its value, a similar approach has been adopted.

An adaptive penalty method has been developed [25] specifically for combinatorial problems solved by TS. This approach makes use of the TS property of short-term memory, i.e., the tabu list. It also makes use of the TS property of long-term memory by incorporating the best solutions found so far (both feasible and infeasible) into the penalty function.

The objective function for infeasible solutions is penalized through the use of a subtractive (Problem P1) or additive (Problem P2) penalty function. The penalty imposed on each constraint violation increases and decreases adaptively over the search according to information stored in the short and long term memories. This is done through the notion of a Near Feasible Threshold (NFT), a boundary just outside of the feasible region. Infeasible solutions are lightly penalized within the NFT region and heavily penalized beyond it. If the search is having difficulty finding good feasible solutions and needs to move closer to the feasible region, then NFT is move closer to feasibility adaptively as the search progresses. On the other hand, if it is desirable to search further through the infeasible region to promote diversity, then NFT is moved further from the feasible boundary. It is important to note that no user intervention is required to update NFT as the search progresses. It is done adaptively based on the relative success of the search using pre-defined rules.

The penalized objective function is based on the unpenalized objective function, the degree of infeasibility and information from the TS short term and long term memory. For Problem P1, the RAP is formulated with two independent constraints (cost and weight) and the objective function is:

$$R_p(t_o;\mathbf{x}) = R(t;\mathbf{x}) - (R_{all} - R_{feas})\left(\left(\frac{\Delta w}{\mathrm{NFT}_w}\right)^{K_1} + \left(\frac{\Delta c}{\mathrm{NFT}_c}\right)^{K_2}\right) \qquad (2)$$

$R_p(t_o;\mathbf{x})$ is the penalized objective function. $R_{all}$ is the unpenalized (feasible or infeasible) system reliability of the best solution found so far, and $R_{feas}$ is the system reliability of the best feasible solution found so far. $\Delta c$ and $\Delta w$ represent the magnitude of the cost and the weight constraint violations. If a constraint is not violated, then $\Delta c$ and/or $\Delta w$ is zero. $K_1$ and $K_2$ are amplification exponents. The amplification exponents, $K_i$, are set to 2 for all work in this paper. However, the method is quite robust to exact value of $K$.

If $R_{all}$ and $R_{feas}$ are equal or similar in value, then the search continues essentially as an unconstrained search because good feasible solutions are being found. Alternatively, if $R_{all}$ is much larger than $R_{feas}$, then the search is having more difficulty in finding good feasible solutions and the penalty is larger to force the search into the feasible region.

For Problem P2, the RAP is formulated with two independent constraints (reliability and weight), the objective function is:

$$C_p(\mathbf{x}) = C(\mathbf{x}) + (C_{feas} - C_{all})\left(\left(\frac{\Delta r}{\mathrm{NFT}_r}\right)^{K_1} + \left(\frac{\Delta w}{\mathrm{NFT}_w}\right)^{K_2}\right) \quad (3)$$

$C_p(\mathbf{x})$ is the penalized objective function. $C_{all}$ is the unpenalized (feasible or infeasible) system cost of the best solution found so far, and $C_{feas}$ is the system cost of the best feasible solution found so far. $\Delta w$ and $\Delta r$ represent the magnitude of the weight and the reliability constraint violations (if any).

$\mathrm{NFT}_{i,j}$ is adaptively updated as the search progresses based on the results encountered in the search. For each constraint, a specific $\mathrm{NFT}_{i,o}$ is initially selected. The tabu list size at any given iteration, $j$, is defined as $T_j$ and the number of feasible solutions on the current tabu list is defined as $F_j$. A feasibility ratio at iteration $j$, $\rho_j$, is defined as:

$$\rho_j = \frac{F_j}{T_j} \quad (4)$$

For constraint $i$, if the current move is to a feasible solution:

$$\mathrm{NFT}_{i,j+1} = \mathrm{NFT}_{i,j}\left(1+\frac{\rho_j}{2}\right) \tag{5}$$

For constraint *i*, if the current move is to an infeasible solution:

$$\mathrm{NFT}_{i,j+1} = \mathrm{NFT}_{i,j}\left(\frac{1+\rho_j}{2}\right) \tag{6}$$

For a given constraint, NFT changes according to the count of the feasible vs. infeasible solutions on the tabu list. The method of Gendreau et al. [33] uses a somewhat similar concept in that the penalty changes with recent constraint violations of the last predefined number of solutions. In Gendreau's method, weight for a constraint that was always violated during the past *h* iterations is multiplied by two. A weight for a constraint that was never violated during the past *h* iterations is divided by two. Otherwise, the weights remain unchanged. This has the property of inflating (deflating) the penalty imposed if the recent search history is entirely within the infeasible (feasible) region. Alternatively, the method based on the $\rho_j$ ratio uses a continuous metric for the feasibility/infeasibility constituency of the tabu list and additionally considers the feasibility of the current move.

Consider the behavior of NFT for a single constraint. If all moves on the tabu list are infeasible and the current move is also infeasible, NFT decreases by a factor of 0.5. This increases the penalty for an infeasible solution and moves the search towards the feasible region. This geometric change in NFT creates a lower bound on NFT of 0. Alternatively, if all moves on the tabu list are feasible and the current move is also feasible, then it may be beneficial to promote search into the infeasible region. In this case, NFT increases by a factor of 1.5. This has the property of encouraging search towards the infeasible region; thereby promoting diversity of prospective solution candidates.

If all moves on the tabu list are feasible and the current move is infeasible, NFT remains unchanged. Similarly, if all moves on the tabu list are infeasible and the current move is feasible, NFT remains unchanged. In these cases, the value of NFT is appropriate as it has moved the search towards the recently unvisited region, either feasible or infeasible. In the next move, if the same region as the last move is chosen, NFT is slightly increased (in the case of recent feasible moves) or slightly decreased (in the case of recent infeasible moves).

An initial value of $\text{NFT}_{i,j}$ needs to be set for each constraint, although the method has been observed to be insensitive to this value. One simple approach is to take a percent of each constraint as an initial value. For example, the initial values, $\text{NFT}_{r,\text{o}}$ and $\text{NFT}_{w,\text{o}}$, are set to 0.1% of $R$ and 50% of $W$ for Problem P2. This formulation can easily handle dynamic tabu list sizes by using the current size, $T_j$. Multiple constraints are handled independently and constraints that are discrete or continuous can be accommodated.

## 3.3 Example Problem

To illustrate the encoding, move operator, evaluation and tabu list, consider a problem with two subsystems, $n_{\max} = 4$ for both subsystems, and the component choices shown in Table 4. If P1 is being solved with $C$=400 and $W$=300, an initial feasible solution might be:

3 7 11 11 5 5 11 11

which consists of one component type 3 and one component type 7 in subsystem one, and two of component type 5 in subsystem two (since there are ten component choices, entry 11 denotes an empty space, or a blank component). In this case, system cost = 320, system weight = 320 and system reliability = 0.882459. Since component mixing is allowed within subsystems, two types of moves are considered. The neighborhood of the first type of move (adding or subtracting the number of a component type) is:

| Solution Encoding (**x**) | | | | | | | | $C$(**x**) | $W$(**x**) | $R$(**x**) |
|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 7 | 11 | 5 | 5 | 11 | 11 | 400 | 352 | 0.948630 |

|  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 11 | 11 | 11 | 5 | 5 | 11 | 11 | 240 | 288 | 0.637383 |
| 3 | 7 | 7 | 11 | 5 | 5 | 11 | 11 | 360 | 418 | 0.941832 |
| 3 | 11 | 11 | 11 | 5 | 5 | 11 | 11 | 280 | 222 | 0.710366 |
| 3 | 7 | 11 | 11 | 5 | 5 | 5 | 11 | 420 | 415 | 0.902850 |
| 3 | 7 | 11 | 11 | 5 | 11 | 11 | 11 | 220 | 225 | 0.758127 |

The second kind of move is to replace a component with another type of component. The neighborhood of this move type is:

| Solution Encoding (**x**) |  |  |  |  |  |  |  | $C$(**x**) | $W$(**x**) | $R$(**x**) |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 11 | 11 | 5 | 5 | 11 | 11 | 335 | 340 | 0.966725 |
| 2 | 7 | 11 | 11 | 5 | 5 | 11 | 11 | 326 | 382 | 0.950611 |
| 4 | 7 | 11 | 11 | 5 | 5 | 11 | 11 | 315 | 380 | 0.879102 |
| 5 | 7 | 11 | 11 | 5 | 5 | 11 | 11 | 301 | 329 | 0.875074 |
| 6 | 7 | 11 | 11 | 5 | 5 | 11 | 11 | 285 | 321 | 0.872052 |
| 7 | 7 | 11 | 11 | 5 | 5 | 11 | 11 | 280 | 386 | 0.857280 |
| 8 | 7 | 11 | 11 | 5 | 5 | 11 | 11 | 276 | 384 | 0.846202 |
| 9 | 7 | 11 | 11 | 5 | 5 | 11 | 11 | 271 | 371 | 0.840159 |
| 10 | 7 | 11 | 11 | 5 | 5 | 11 | 11 | 266 | 354 | 0.755557 |
| 11 | 7 | 11 | 11 | 5 | 5 | 11 | 11 | 240 | 288 | 0.637383 |
| 3 | 1 | 11 | 11 | 5 | 5 | 11 | 11 | 375 | 274 | 0.968112 |
| 3 | 2 | 11 | 11 | 5 | 5 | 11 | 11 | 366 | 316 | 0.955501 |
| 3 | 3 | 11 | 11 | 5 | 5 | 11 | 11 | 360 | 254 | 0.902165 |
| 3 | 4 | 11 | 11 | 5 | 5 | 11 | 11 | 355 | 314 | 0.899537 |
| 3 | 5 | 11 | 11 | 5 | 5 | 11 | 11 | 341 | 263 | 0.896384 |
| 3 | 6 | 11 | 11 | 5 | 5 | 11 | 11 | 325 | 255 | 0.894020 |
| 3 | 8 | 11 | 11 | 5 | 5 | 11 | 11 | 316 | 318 | 0.873789 |
| 3 | 9 | 11 | 11 | 5 | 5 | 11 | 11 | 311 | 305 | 0.869060 |
| 3 | 10 | 11 | 11 | 5 | 5 | 11 | 11 | 306 | 288 | 0.802850 |
| 3 | 11 | 11 | 11 | 5 | 5 | 11 | 11 | 280 | 222 | 0.710366 |
| 3 | 7 | 11 | 11 | 1 | 5 | 11 | 11 | 357 | 308 | 0.896588 |
| 3 | 7 | 11 | 11 | 2 | 5 | 11 | 11 | 352 | 321 | 0.894506 |
| 3 | 7 | 11 | 11 | 3 | 5 | 11 | 11 | 347 | 319 | 0.889747 |
| 3 | 7 | 11 | 11 | 4 | 5 | 11 | 11 | 342 | 318 | 0.885583 |
| 3 | 7 | 11 | 11 | 6 | 5 | 11 | 11 | 279 | 288 | 0.878741 |
| 3 | 7 | 11 | 11 | 7 | 5 | 11 | 11 | 274 | 290 | 0.849145 |
| 3 | 7 | 11 | 11 | 8 | 5 | 11 | 11 | 261 | 274 | 0.822375 |
| 3 | 7 | 11 | 11 | 9 | 5 | 11 | 11 | 256 | 258 | 0.815980 |
| 3 | 7 | 11 | 11 | 10 | 5 | 11 | 11 | 250 | 276 | 0.808544 |
| 3 | 7 | 11 | 11 | 11 | 5 | 11 | 11 | 220 | 225 | 0.758127 |
| 3 | 7 | 11 | 11 | 5 | 1 | 11 | 11 | 357 | 308 | 0.896588 |
| 3 | 7 | 11 | 11 | 5 | 2 | 11 | 11 | 352 | 321 | 0.894506 |
| 3 | 7 | 11 | 11 | 5 | 3 | 11 | 11 | 347 | 319 | 0.889747 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 11 | 11 | 5 | 4 | 11 | 11 | 342 | 318 | 0.885583 |
| 3 | 7 | 11 | 11 | 5 | 6 | 11 | 11 | 279 | 288 | 0.878741 |
| 3 | 7 | 11 | 11 | 5 | 7 | 11 | 11 | 274 | 290 | 0.849145 |
| 3 | 7 | 11 | 11 | 5 | 8 | 11 | 11 | 261 | 274 | 0.822375 |
| 3 | 7 | 11 | 11 | 5 | 9 | 11 | 11 | 256 | 258 | 0.815980 |
| 3 | 7 | 11 | 11 | 5 | 10 | 11 | 11 | 250 | 276 | 0.808544 |
| 3 | 7 | 11 | 11 | 5 | 11 | 11 | 11 | 220 | 225 | 0.758127 |

Among the solutions of both neighborhoods, the best one is

3 1 11 11 5 5 11 11

Therefore, a move is made to this solution.

The tabu list is now updated to reflect the most recent move. The tabu list is composed of the most recent subsystem configuration that has been changed. In the example, the first subsystem was previously composed of components 3 and 7 in parallel. The best move was to change it to components 1 and 3 in parallel. Specifically, tabu list entries include the subsystem number, the component selections (including blanks), and the system cost and weight, which are necessary to denote the feasibility of the solution. For the example move, the following entry is added to the tabu list.

1 | 3 1 11 11 | 375 | 274

## 4. TEST PROBLEMS AND RESULTS

TSRAP was demonstrated and evaluated using three general problems and a total of 72 variations. The results were compared to genetic algorithm and exact solutions using an IP model. The results indicate that the TS methodology offers advantages over the other approaches.

### 4.1 Test Problem 1

The first test problems used to demonstrate TSRAP for Problem P1 were originally proposed by Fyffe et al. [15] and modified by Nakagawa & Miyazaki [16]. Fyffe et al. specified constraint limits of 130 units of system cost, 170 units of system weight and $k_i = 1$ (i.e., 1-out-of-*n*:G subsystems). Nakagawa & Miyazaki developed 33 variations of the original problem, where the cost constraint is maintained at 130 and the weight constraint varies from 191 to 159. The component cost, weight and reliability values were originally presented in Fyffe [15] and they are not reproduced here.

Earlier approaches to the problem determined optimal solutions through dynamic programming and IP models, but only a restricted set of solutions was considered due to computational or formulation limitations of exact solution methods. Coit & Smith [2] solved this problem with a GA without restricting the search space. These are the results used as a basis for TSRAP and GA comparisons.

TSRAP was applied 10 times with different starting solutions. The results are given in Table 1. The best solution found among the 10 runs is presented as well as the standard deviation of the 10 final solutions. The standard deviation is an important measure of the robustness of the search to the starting solution. The table also presents the corresponding results from the GA [2] where 10 runs of the algorithm were also performed. In this table, the maximum possible improvement (MPI), is the percent that one solution improves upon another considering that reliability is bounded by one.

As the results in Table 1 indicate, TSRAP generally outperformed the GA. TSRAP obtained a higher reliability in 22 of 33 test cases. Additionally, it had a lower standard deviation in 21 of 33 test cases. Thus, TSRAP generally yielded solutions with higher reliability and was more consistent. Since GA contains many more stochastic elements, it is not surprising that TSRAP is more consistent across initial solution.

A comparison of algorithm efficiency is often made based on computer CPU time. That would not be meaningful for this comparison because the algorithms were run on different computers (mainframes, PCs) with different operating systems and processors. Alternatively, a comparison can be made based on the number of objective function evaluations required, as it is this calculation which takes most of the CPU time. In many respects, this is more meaningful because it provides an absolute measure that maintains relevance as computer processing time continues to decrease. For the GA, the stopping criterion was always 1,200 and every GA run had 48,040 function evaluations. The TSRAP stopping criterion was based on the number of moves without improvement so it varied.

Table 1 displays the average number of function evaluations for each of the 33 test problems. While these average numbers are consistently larger than the corresponding number of GA function evaluations, they actually do provide evidence that TSRAP is more efficient. Each move within TSRAP required the re-calculation of only one (of 14) subsystems, while every new child or mutant solutions from the GA required the calculation of all 14 subsystems. Thus, each function evaluation within TSRAP requires approximately 14 times less computation time. In reviewing the number of solutions searched between GA and TSRAP and the reduction of TSRAP per solution, TSRAP reduced computation time approximately 40% over the test suite. While these results are encouraging, they can not necessarily be generalized to all RAP problem formulations, particularly considering the different stopping criterion.

## 4.2 Test Problem 2

Coit & Liu [27] proposed a problem to optimize the system reliability of a system with 14 $k$-out-of-$n$:G subsystems with $k_i \in \{1,2,3\}$ and exponential distributed failure times. Their problem is a modified version of the 33 problem variations originally given by Nakagawa & Miyazaki [16]. For each of 14 subsystems, component choices with cost, weight and exponential distribution parameter ($\lambda_{ij}$) are given in Table 2. The objective is to maximize system reliability at the time of 100 hours subject to given cost and weight constraints, $C$ and $W$, respectively. They solved the problem for a restricted search space so exact optimization could be used.

A primary advantage of TSRAP for problems of this type are that the restriction, that disallows mixing of component types within a subsystem, is not necessary. Thus, potentially better solutions, with higher system reliability can be found. Nevertheless, the same restriction can be imposed on TSRAP to allow a direct comparison. For the restricted search space, the IP results are known to be optimal so a direct comparison of results provides a credible indication of TSRAP performance. (Note that a comparison of computational effort is not germane to this problem. When an exact method, such as IP, can be used for design, it should be preferred as optimality will be assured. However, as search space grows, the computational effort required by exact methods becomes prohibitive. That is why only the comparison that does not allow mixing can be made here. If mixing were allowed, it would be impossible to run to completion.)

The 33 problem variations from Coit & Liu [27] were solved based on active redundancy for all subsystems. The results of TSRAP with and without component mixing within subsystem and the corresponding IP results are presented in Table 2. Ten runs of TSRAP were performed for each problem with different initial solutions. The results without component mixing are the same as the optimum results for all 33 cases. This is very encouraging because it provides an indication that TSRAP is providing optimal solutions. For many other problems, as in test problem 1, the optimal solution is not known or cannot be calculated because of computational limits.

When component mixing is allowed in the subsystems, the system reliability improves in 25 of the 33 variations. This demonstrates the value of expanding the search space to consider different components within a single subsystem. While the improvements might seem very small, recall that these apply to system operation. If, for an example, we take a hypothetical fleet of 60 767s. A mission time of eight hours might be selected as a typical overseas flight. (Fyffe et al. [15] nor Nakagawa & Miyazaki [16] stated a mission time.) If each aircraft makes six overseas flights per week, an increase in system reliability of 0.00308, as achieved in the last instance, would translate to an expected 58 fewer failures over the course of a year (60x6x52x.00308). Also, when system failure is of high consequence, as in a nuclear power plant for instance, that any improvement in reliability is worthwhile.

## 4.3 Test Problem 3

TSRAP performance for Problem P2 was studied by using the corresponding example problem from Coit & Smith [2]. The component choices are shown in Table 4. This sample problem was designed with two subsystems where $k_1$ is 4 and $k_2$ is 2 (with $n_{max,i} = 8$). For each subsystem, there are 10 component choices with different cost, weight, and reliability values. This problem is difficult in several respects since both subsystems are $k$-out-of-$n$:G redundancy with $k > 1$. Also for each subsystem, there are ten different component choices available, allowing for numerous design possibilities.

By changing the reliability and weight constraints, six different cases were defined as shown in Table 5. Twenty runs of TSRAP were performed and compared with the previous GA results [2] in Table 6 along with the optimal solutions found by enumeration. The meta-heuristics perform very similarly and computational comparisons are similar to those reported in section 3.1 with TS requiring more solutions, but gaining the considerable efficiency of recalculating reliability using only one subsystem. Again, allowing component mixing in subsystems yields costs which are better than that which could be obtained by using any of the previously presented formulations.

## 5. CONCLUSIONS

TS has previously been demonstrated to be a successful optimization approach for many diverse problem domains. However, its ability to provide sound solutions to the RAP had not previously been reported. Within this paper, a TS is described that is designed for the RAP with the use of a penalty function to allow, and even promote, search in the infeasible region with an adaptively changing NFT. TSRAP was demonstrated on three test problems with encouraging results. Like GA, the TS methodology can be applied to many diverse reliability optimization problems where mathematical programming approaches have not been successful. When compared to GA, TSRAP results in superior performance in terms of best solutions found and reduced variability and greater efficiency. The TS reported herein is rather simple, that is, some features normally used effectively in complex problems such as candidate lists and long term memory strategies, are not incorporated. There are opportunities for improved effectiveness and efficiency by considering the addition of these features to the TS devised here.

## REFERENCES

[1] W. Kuo and V. R. Prasad, An annotated overview of system-reliability optimization, *IEEE Transactions on Reliability*, 49/2 (2000), 176-187.

[2] D. W. Coit and A. E. Smith, Reliability optimization of series-parallel systems using a genetic algorithm, *IEEE Transactions on Reliability*, 45/2 (1996), 254-260.

[3] F. Glover, Future paths for integer programming and links to artificial intelligence, *Computers & Operations Research*, 13 (1986), 533-549.

[4] F. Glover, Tabu Search-Part I, *ORSA Journal of Computing*, 1 (1989), 190-206.

[5] F. Glover, Tabu Search-Part II, *ORSA Journal of Computing*, 2 (1990), 4-32.

[6] F. Glover, E. Taillard, and D. de Werra, A user's guide to tabu search, *Annals of Operations Research*, 41 (1993), 3-28.

[7] F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, London, UK (1997).

[8] C. R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*, John Wiley & Sons, New York (1993).

[9] F. Glover, M. Laguna and R. Marti, Fundamentals of scatter search and path relinking, *Control and Cybernetics*, 29/3 (2000), 653-684.

[10] M. S. Chern, On the computational complexity of reliability redundancy allocation in a series system, *Operations Research Letters*, 11 (1992), 309-315.

[11] F. A. Tillman, C. -L. Hwang, and W. Kuo, Optimization techniques for system reliability with redundancy – A review, *IEEE Transactions on Reliability*, R-26/3 (1977), 148-155.

[12] R. E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[13] R. E. Bellman and E. Dreyfus, Dynamic programming and reliability of multicomponent devices, *Operations Research*, 6 (1958), 200-206.

[14] R. E. Bellman and E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, Princeton, NJ, 1962.

[15] D. E. Fyffe, W. W. Hines, and N. K. Lee, System reliability allocation and a computational algorithm, *IEEE Transactions on Reliability*, R-17 (1968), 74-79.

[16] Y. Nakagawa and S. Miyazaki, Surrogate constraints algorithm for reliability optimization problems with two constraints, *IEEE Transactions on Reliability*, R-30 (1981), 175-180.

[17] M. Ghare and R. E. Taylor, Optimal redundancy for reliability in series system, *Operations Research*, 17 (1969), 838-847.

[18] R. L. Bulfin and C. Y. Liu, Optimal allocation of redundant components for large systems, *IEEE Transactions on Reliability*, R-34 (1985), 241-247.

[19] K. B. Misra and U. Sharma, An efficient algorithm to solve integer programming problems arising in system reliability design, *IEEE Transactions on Reliability*, 40 (1991), 81-91.

[20] M. Gen, K. Ida, Y. Tsujimura, and C. E. Kim, Large scale 0-1 fuzzy goal programming and its application to reliability optimization problem, *Computers & Industrial Engineering*, 24

(1993), 539-549.

[21] F. A. Tillman, C. -L. Hwang, and W. Kuo, Determining component reliability and redundancy for optimum system reliability, *IEEE Transactions on Reliability*, R-26/3 (1977), 162-165.

[22] L. Painton and J. Campbell, Genetic algorithms in optimization of system reliability, *IEEE Transactions on Reliability*, 44/2 (1995), 172-179.

[23] D. W. Coit and A. E. Smith, Penalty guided genetic search for reliability design optimization, *Computers & Industrial Engineering*, 30/4 (1996), 895-904.

[24] D. W. Coit, A. E. Smith, and D. M. Tate, Adaptive penalty methods for genetic optimization of constrained combinatorial problems, *INFORMS Journal on Computing*, 8 (1996), 173-182.

[25] S. Kulturel-Konak, B. A. Norman, D. W. Coit, and A. E. Smith, Exploiting tabu search memory in constrained problems, *INFORMS Journal on Computing*, in revision.

[26] Y. C. Liang and A. E. Smith, An ant system approach to redundancy allocation, *Proceedings of the 1999 Congress on Evolutionary Computation*, (1999), 1478-1484.

[27] D. W. Coit and J. Liu, System reliability optimization with k-out-of-n subsystems, *International Journal of Reliability, Quality and Safety Engineering*, 7/2, (2000), 129-143.

[28] J. A. Bland, Memory-based technique for optimal structural design, *Engineering Applications of Artificial Intelligence*, 11/3, (1998), 319-325.

[29] J. A. Bland, Structural design optimization with reliability constraints using tabu search, *Engineering Optimization*, 30/1 (1998), 55-74.

[30] J. Xu, S. Y. Chiu and F. Glover, Optimizing a ring-based private line telecommunication network using Tabu Search, *Management Science*, 45/3 (1999) 330-345.

[31] S. Pierre and A. Elgibaoui, Tabu-search approach for designing computer-network topologies with unreliable components, *IEEE Transactions on Reliability*, 46/3 (1997) 350-359.

[32] R. R. Brooks, S. S. Iyengar and S. Rai, Minimizing cost of redundant sensor-systems with non-monotone and monotone search algorithms, *Proceedings of the Annual Reliability and Maintainability Symposium*, (1997), 307-313.

[33] M. Gendreau, A. Hertz, and G. Laporte, A tabu search heuristic for the vehicle routing problems, *Management Science*, 40 (1994), 1276-1290.

Table 1. Test Problem 1 - Comparison of GA [2] and TSRAP.

| | | | GA [2] – 10 runs | | TSRAP – 10 runs | | | |
|---|---|---|---|---|---|---|---|---|
| No | *C* | *W* | max *R* | Std Dev | max *R* | Std Dev | average evaluations | %MPI |
| 1 | 130 | 191 | 0.98670 | 0.000422 | 0.986811 | 0.000644 | 347,969 | 0.83% |
| 2 | 130 | 190 | 0.98570 | 0.000595 | 0.986416 | 0.000000 | 503,369 | 5.01% |
| 3 | 130 | 189 | 0.98560 | 0.000853 | 0.985922 | 0.000000 | 484,283 | 2.24% |
| 4 | 130 | 188 | 0.98500 | 0.000306 | 0.985378 | 0.000765 | 387,029 | 2.52% |
| 5 | 130 | 187 | 0.98440 | 0.000748 | 0.984688 | 0.000307 | 347,116 | 1.85% |
| 6 | 130 | 186 | 0.98360 | 0.000586 | 0.984176 | 0.000386 | 413,235 | 3.51% |
| 7 | 130 | 185 | 0.98310 | 0.000856 | 0.983505 | 0.000510 | 366,652 | 2.40% |
| 8 | 130 | 184 | 0.98230 | 0.000465 | 0.982994 | 0.000450 | 367,142 | 3.92% |
| 9 | 130 | 183 | 0.98190 | 0.000389 | 0.982256 | 0.000628 | 361,935 | 1.97% |
| 10 | 130 | 182 | 0.98110 | 0.000381 | 0.981518 | 0.000428 | 402,144 | 2.21% |
| 11 | 130 | 181 | 0.98020 | 0.000842 | 0.981027 | 0.000356 | 367,933 | 4.18% |
| 12 | 130 | 180 | 0.97970 | 0.000791 | 0.980290 | 0.000469 | 436,098 | 2.91% |
| 13 | 130 | 179 | 0.97910 | 0.000538 | 0.979505 | 0.000341 | 303,575 | 1.94% |
| 14 | 130 | 178 | 0.97830 | 0.000701 | 0.978400 | 0.000143 | 262,093 | 0.46% |
| 15 | 130 | 177 | 0.97720 | 0.001031 | 0.977474 | 0.000198 | 344,122 | 1.20% |
| 16 | 130 | 176 | 0.97640 | 0.000751 | 0.976690 | 0.000669 | 301,944 | 1.23% |
| 17 | 130 | 175 | 0.97530 | 0.000795 | 0.975708 | 0.000288 | 345,226 | 1.65% |
| 18 | 130 | 174 | 0.97435 | 0.000812 | 0.974788 | 0.000610 | 276,510 | 1.71% |
| 19 | 130 | 173 | 0.97362 | 0.000753 | 0.973827 | 0.000534 | 260,570 | 0.78% |
| 20 | 130 | 172 | 0.97266 | 0.001083 | 0.973027 | 0.000166 | 355,909 | 1.34% |
| 21 | 130 | 171 | 0.97186 | 0.000812 | 0.971929 | 0.000000 | 334,564 | 0.25% |
| 22 | 130 | 170 | 0.97076 | 0.000821 | 0.970760 | 0.000490 | 311,927 | 0.00% |
| 23 | 130 | 169 | 0.96922 | 0.000415 | 0.969291 | 0.000351 | 310,691 | 0.23% |
| 24 | 130 | 168 | 0.96813 | 0.000596 | 0.968125 | 0.000957 | 320,568 | -0.02% |
| 25 | 130 | 167 | 0.96634 | 0.000304 | 0.966335 | 0.000739 | 395,987 | -0.01% |
| 26 | 130 | 166 | 0.96504 | 0.000569 | 0.965042 | 0.000616 | 322,723 | 0.01% |
| 27 | 130 | 165 | 0.96371 | 0.000474 | 0.963712 | 0.000932 | 315,828 | 0.01% |
| 28 | 130 | 164 | 0.96242 | 0.000659 | 0.962422 | 0.001044 | 413,602 | 0.01% |
| 29 | 130 | 163 | 0.96064 | 0.000401 | 0.959980 | 0.000176 | 347,964 | -1.68% |
| 30 | 130 | 162 | 0.95912 | 0.000833 | 0.958205 | 0.000053 | 305,090 | -2.24% |
| 31 | 130 | 161 | 0.95803 | 0.000808 | 0.956922 | 0.001230 | 256,383 | -2.64% |
| 32 | 130 | 160 | 0.95567 | 0.000473 | 0.955604 | 0.001424 | 343,186 | -0.15% |
| 33 | 130 | 159 | 0.95432 | 0.000363 | 0.954325 | 0.002004 | 346,894 | 0.01% |

%MPI = 100% * (TSRAP max - GA max) / (1 - GA max)

Table 2. Component Data for Test Problem 2.

| sub system | | component choices | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | | | 2 | | | 3 | | | 4 | | |
| $i$ | $k_i$ | $\lambda_{1j}$* | $c_{1j}$ | $w_{1j}$ | $\lambda_{2j}$ | $c_{2j}$ | $w_{2j}$ | $\lambda_{3j}$ | $c_{3j}$ | $w_{3j}$ | $\lambda_{4j}$ | $c_{4j}$ | $w_{4j}$ |
| 1 | 1 | .001054 | 1 | 3 | .000726 | 1 | 4 | .000943 | 2 | 2 | .000513 | 2 | 5 |
| 2 | 2 | .000513 | 2 | 8 | .000619 | 1 | 10 | .000726 | 1 | 9 | - - | | |
| 3 | 1 | .001625 | 2 | 7 | .001054 | 3 | 5 | .001393 | 1 | 6 | .000834 | 4 | 4 |
| 4 | 2 | .001863 | 3 | 5 | .001393 | 4 | 6 | .001625 | 5 | 4 | - - | | |
| 5 | 1 | .000619 | 2 | 4 | .000726 | 2 | 3 | .000513 | 3 | 5 | - - | | |
| 6 | 2 | .000101 | 3 | 5 | .000202 | 3 | 4 | .000305 | 2 | 5 | .000408 | 2 | 4 |
| 7 | 1 | .000943 | 4 | 7 | .000834 | 4 | 8 | .000619 | 5 | 9 | - - | | |
| 8 | 2 | .002107 | 3 | 4 | .001054 | 5 | 7 | .000943 | 6 | 6 | - - | | |
| 9 | 3 | .000305 | 2 | 8 | .000101 | 3 | 9 | .000408 | 4 | 7 | .000943 | 3 | 8 |
| 10 | 3 | .001863 | 4 | 6 | .001625 | 4 | 5 | .001054 | 5 | 6 | - - | | |
| 11 | 3 | .000619 | 3 | 5 | .000513 | 4 | 6 | .000408 | 5 | 6 | - - | | |
| 12 | 1 | .002357 | 2 | 4 | .001985 | 3 | 5 | .001625 | 4 | 6 | .001054 | 5 | 7 |
| 13 | 2 | .000202 | 2 | 5 | .000101 | 3 | 5 | .000305 | 2 | 6 | - - | | |
| 14 | 3 | .001054 | 4 | 6 | .000834 | 4 | 7 | .000513 | 5 | 6 | .000101 | 6 | 9 |

*units for $\lambda_{ij}$ are failures/hour

Table 3. Test Problem 2 - Comparisons of IP and TSRAP.

| | | | IP | TSRAP (no mixing) - 10 runs | | | TSRAP (mixing) -10 runs | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Max | Std | average | Max | Std | average | |
| No | *C* | *W* | *R* | *R* | Dev | evaluations | *R* | Dev | evaluations | %MPI |
| 1 | 130 | 191 | 0.60665 | 0.60665 | 0.000000 | 83876 | 0.60766 | 0.000000 | 343333 | 0.26% |
| 2 | 130 | 190 | 0.59064 | 0.59064 | 0.000583 | 123766 | 0.59538 | 0.000610 | 377963 | 1.16% |
| 3 | 130 | 189 | 0.58105 | 0.58105 | 0.000345 | 91708 | 0.58827 | 0.000000 | 368893 | 1.72% |
| 4 | 130 | 188 | 0.57327 | 0.57327 | 0.001371 | 96616 | 0.57734 | 0.001508 | 289590 | 0.95% |
| 5 | 130 | 187 | 0.56250 | 0.56250 | 0.000491 | 106152 | 0.57044 | 0.002255 | 265562 | 1.81% |
| 6 | 130 | 186 | 0.55498 | 0.55498 | 0.000000 | 90123 | 0.55892 | 0.000910 | 292436 | 0.88% |
| 7 | 130 | 185 | 0.54285 | 0.54285 | 0.000698 | 105121 | 0.54936 | 0.000523 | 279319 | 1.42% |
| 8 | 130 | 184 | 0.53401 | 0.53401 | 0.000238 | 83484 | 0.54112 | 0.001718 | 287372 | 1.53% |
| 9 | 130 | 183 | 0.52981 | 0.52981 | 0.000000 | 104231 | 0.53226 | 0.000194 | 367116 | 0.52% |
| 10 | 130 | 182 | 0.52272 | 0.52272 | 0.000000 | 95722 | 0.52385 | 0.002439 | 282638 | 0.24% |
| 11 | 130 | 181 | 0.51290 | 0.51290 | 0.000549 | 86564 | 0.51528 | 0.002924 | 311123 | 0.49% |
| 12 | 130 | 180 | 0.50604 | 0.50604 | 0.000000 | 106757 | 0.50604 | 0.002000 | 232905 | **0.00%** |
| 13 | 130 | 179 | 0.49057 | 0.49057 | 0.001899 | 85898 | 0.49402 | 0.002297 | 263555 | 0.68% |
| 14 | 130 | 178 | 0.48468 | 0.48468 | 0.000000 | 90370 | 0.48573 | 0.001836 | 291947 | 0.20% |
| 15 | 130 | 177 | 0.47491 | 0.47491 | 0.000277 | 109807 | 0.47778 | 0.004778 | 290110 | 0.55% |
| 16 | 130 | 176 | 0.46922 | 0.46922 | 0.000000 | 85526 | 0.46922 | 0.005300 | 258560 | **0.00%** |
| 17 | 130 | 175 | 0.45298 | 0.45298 | 0.000000 | 94737 | 0.45743 | 0.000327 | 222826 | 0.81% |
| 18 | 130 | 174 | 0.44878 | 0.44878 | 0.001711 | 116496 | 0.44975 | 0.006295 | 243874 | 0.18% |
| 19 | 130 | 173 | 0.43923 | 0.43923 | 0.000226 | 97602 | 0.44239 | 0.005074 | 312906 | 0.56% |
| 20 | 130 | 172 | 0.43446 | 0.43446 | 0.000000 | 98716 | 0.43446 | 0.005173 | 266489 | **0.00%** |
| 21 | 130 | 171 | 0.41942 | 0.41942 | 0.001902 | 97759 | 0.42033 | 0.003418 | 255287 | 0.16% |
| 22 | 130 | 170 | 0.41050 | 0.41050 | 0.000030 | 107748 | 0.41345 | 0.002625 | 245476 | 0.50% |
| 23 | 130 | 169 | 0.40604 | 0.40604 | 0.003841 | 72328 | 0.40604 | 0.004025 | 220822 | **0.00%** |
| 24 | 130 | 168 | 0.39025 | 0.39025 | 0.000000 | 88613 | 0.39109 | 0.000000 | 287080 | 0.14% |
| 25 | 130 | 167 | 0.38194 | 0.38194 | 0.000000 | 80788 | 0.38469 | 0.000000 | 244641 | 0.44% |
| 26 | 130 | 166 | 0.37779 | 0.37779 | 0.000000 | 117293 | 0.37779 | 0.000000 | 251777 | **0.00%** |
| 27 | 130 | 165 | 0.36472 | 0.36472 | 0.000000 | 83956 | 0.36550 | 0.000729 | 263279 | 0.12% |
| 28 | 130 | 164 | 0.35696 | 0.35696 | 0.001869 | 85885 | 0.35952 | 0.000000 | 309496 | 0.40% |
| 29 | 130 | 163 | 0.35308 | 0.35308 | 0.000000 | 100222 | 0.35308 | 0.000000 | 241806 | **0.00%** |
| 30 | 130 | 162 | 0.33243 | 0.33243 | 0.000000 | 103432 | 0.33243 | 0.001725 | 243964 | **0.00%** |
| 31 | 130 | 161 | 0.32392 | 0.32392 | 0.000000 | 97440 | 0.32392 | 0.002312 | 275075 | **0.00%** |
| 32 | 130 | 160 | 0.30908 | 0.30908 | 0.001085 | 96723 | 0.31209 | 0.000000 | 309487 | 0.43% |
| 33 | 130 | 159 | 0.30250 | 0.30250 | 0.009224 | 75507 | 0.30558 | 0.000000 | 230889 | 0.44% |

%MPI = 100% * (TSRAP max rel. - IP rel.) / (1 - IP rel.)

Table 4. Component Data for Test Problem 3.

| component choice ($j$) | Subsystem ($i$) | | | | | |
|---|---|---|---|---|---|---|
| | 1 ($k_1 = 4$) | | | 2 ($k_2 = 2$) | | |
| | $r_{1j}$ | $c_{1j}$ | $w_{1j}$ | $r_{2j}$ | $c_{2j}$ | $w_{2j}$ |
| 1 | 0.981 | 95 | 52 | 0.931 | 137 | 83 |
| 2 | 0.933 | 86 | 94 | 0.917 | 132 | 96 |
| 3 | 0.730 | 80 | 32 | 0.885 | 127 | 94 |
| 4 | 0.720 | 75 | 92 | 0.857 | 122 | 93 |
| 5 | 0.708 | 61 | 41 | 0.836 | 100 | 95 |
| 6 | 0.699 | 45 | 33 | 0.811 | 59 | 63 |
| 7 | 0.655 | 40 | 98 | 0.612 | 54 | 65 |
| 8 | 0.622 | 36 | 96 | 0.432 | 41 | 49 |
| 9 | 0.604 | 31 | 83 | 0.389 | 36 | 33 |
| 10 | 0.352 | 26 | 66 | 0.339 | 30 | 51 |

Table 5. Comparison of GA and TSRAP for Test Problem 3.

| Problem Description | | | | | C&S GA [2] - 20 runs | | | | TSRAP - 20 runs | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Case | $R$ | $W$ | Global Minimum | Previous Best* | Minimum Cost | Average Cost | Number Optimal | Number Feasible | Minimum Cost | Average Cost | Number Optimal | Number Feasible |
| 1 | 0.98 | 650 | 727 | 770 | 727 | 727.25 | 18/20 | 20/20 | 727 | 727.80 | 18/20 | 20/20 |
| 2 | 0.98 | 600 | 736 | 770 | 736 | 736.90 | 11/20 | 20/20 | 736 | 737.00 | 12/20 | 20/20 |
| 3 | 0.98 | 550 | 747 | 871 | 747 | 747.00 | 20/20 | 20/20 | 747 | 749.35 | 19/20 | 20/20 |
| 4 | 0.95 | 600 | 656 | 711 | 656 | 656.00 | 20/20 | 20/20 | 656 | 658.10 | 18/20 | 20/20 |
| 5 | 0.95 | 550 | 661 | 711 | 661 | 661.00 | 20/20 | 20/20 | 661 | 661.00 | 20/20 | 20/20 |
| 6 | 0.95 | 500 | 661 | none | 661 | 680.80 | 18/20 | 20/20 | 661 | 661.00 | 20/20 | 20/20 |

*The lowest minimum cost could be found by N&M [16] and Bulfin and Liu [18] formulations

Table 6. Optimal Solutions for Test Problem 3.

| | | | | Component Choices | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | subsystem 1($k$ = 4) | | | | subsystem 2 ($k$ = 2) | | |
| Case | Cost | Reliability | Weight | 1 | 6 | 7 | 8 | 6 | 9 | 10 |
| 1 | 727 | 0.975 | 640 | 4 | 1 | 0 | 1 | 4 | 0 | 1 |
| 2 | 736 | 0.9768 | 577 | 4 | 2 | 0 | 0 | 4 | 0 | 1 |
| 3 | 747 | 0.9819 | 545 | 5 | 0 | 0 | 0 | 4 | 1 | 0 |
| 4 | 656 | 0.9506 | 558 | 4 | 0 | 1 | 0 | 4 | 0 | 0 |
| 5 | 661 | 0.9537 | 661 | 4 | 1 | 0 | 0 | 4 | 0 | 0 |
| 6 | 661 | 0.9537 | 661 | 4 | 1 | 0 | 0 | 4 | 0 | 0 |

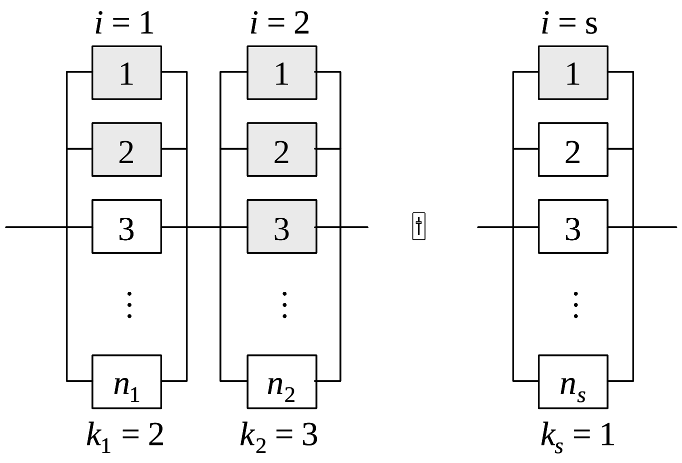

Figure 1: System Configuration with *k*-out-of-*n* Subsystems.