IIE Transactions (2003) **35**, 515–526
Copyright © 2003 "IIE"
0740-817X/03 $12.00+.00
DOI: 10.1080/07408170390193044

Efficiently solving the redundancy allocation problem using tabu search

SADAN KULTUREL-KONAK[1], ALICE E. SMITH[2] and DAVID W. COIT[3]

[1]*Management Information Systems, Penn State Berks-Lehigh Valley College, Tulpehocken Road, P.O. Box 7009, Reading, PA 19610-6009, USA*
E-mail: sadan@psu.edu
[2]*Department of Industrial and Systems Engineering, Auburn University, 207 Dunstan Hall, Auburn, AL 36849, USA*
E-mail: aesmith@eng.auburn.edu
[3]*Department of Industrial and Systems Engineering, Rutgers University, 96 Frelinghuysen Road, Piscataway, NJ 08554, USA*
E-mail: coit@rci.rutgers.edu

Received August 2001 and accepted August 2002

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.

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 (Kuo and Prasad, 2000). 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 and Smith (1996a).

Genetic Algorithms (GAs) have been able to overcome some of the deficiencies of mathematical programming approaches, and for many problems, the use of GAs has provided excellent results. However, a 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 (1986). 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 including Glover (1989, 1990), Glover *et al.* (1993), and Reeves (1993) the most comprehensive source, however, being Glover and Laguna (1997).

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 *et al.* (2000).)

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 recalculation 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 recalculation of the system reliability.

The following notation will be used throughout the remainder of the paper.

Notation

$R(t, \mathbf{x})$	=	system reliability at time t, depending on $\mathbf{x}$;
$\mathbf{x}$	=	$(x_{11}, x_{12}, \ldots, x_{1,m_1}, x_{21}, x_{22}, \ldots, x_{2,m_2}, x_{31}, \ldots, x_{s,m_s})^T$;
x_{ij}	=	quantity of the jth available component used in subsystem i;
m_i	=	number of available components for subsystem i;
s	=	number of subsystems;
$\mathbf{n}$	=	$(n_1, n_2, \ldots, n_s)$;
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 ($n_i \leq n_{\max,i} \forall i$);
$C(\mathbf{x})$	=	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 jth available component for subsystem i;
t_o	=	mission time (fixed);
$\mathbf{k}$	=	$(k_1, k_2, \ldots, k_s)$;
k_i	=	minimum number of operating components required for subsystem i;
λ_{ij}	=	parameter for exponential distribution, $f_{ij}(t) = \lambda_{ij} \exp(-\lambda_{ij} t)$;
$\mathrm{NFT}_{i,j}$	=	Near Feasible Threshold for the ith constraint at the jth move of the TS;
F_j	=	number of feasible solutions on the tabu list;
T_j	=	total number of solutions on the tabu list;
ρ_j	=	feasibility ratio, $\rho_j = F_j / T_j$.

2. The RAP

The most studied design configuration of the RAP is a series system of s independent k-out-of-n:G subsystems (Fig. 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

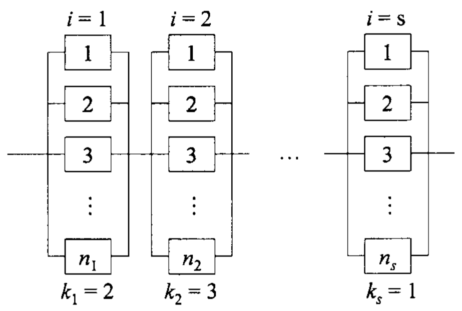

Fig. 1. System configuration with k-out-of-n subsystem.

NP-hard (Chern, 1992) and has been studied in many different forms. An overview and summary of work for different approaches to RAP is presented in Tillman *et al.* (1997a), and more recently by Kuo and Prasad (2000).

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 the system reliability given overall restrictions on the system cost, C, and the system weight, W. Problem (P2) minimizes the system cost given overall restrictions on the maximum system weight, W, and the 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 also more complex functions.

Problem (P1): reliability maximization

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

subject to

$$\sum_{i=1}^{s}\sum_{j=1}^{m_i} c_{ij}x_{ij} \leq C,$$

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

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

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

Problem (P2): cost minimization

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

subject to

$$R(t_o; \mathbf{x}) \geq R,$$

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

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

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

The series-parallel RAP (i.e., $k_i = 1\ \forall i$) has been widely studied. The different approaches include dynamic programming (Bellman, 1957; Bellman and Dreyfus, 1958, 1962; Fyffe *et al.* 1968; Nakagawa and Miyazaki, 1981), IP (Ghare and Taylor, 1969; Bulfin and Liu, 1985; Misra and Sharma, 1991; Gen *et al.* 1993), and mixed-integer and nonlinear programming (Tillman *et al.*, 1977b). More recently, meta-heuristic methods such as GA (Painton and Campbell, 1995; Coit and Smith, 1996a, 1996b; Coit *et al.* 1996), TS (Kulturel-Konak *et al.*, 2003), and the Ant System Algorithm (Liang and Smith, 1999) have been applied to the problem.

An IP approach was used by Coit and Liu (2000) 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 zero-one 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 zero-one 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:

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

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 GAs. According to Kuo and Prasad (2000), "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 (1998a, 1998b), and telecommunications network design problems with unreliable components by Pierre and Elgibaoui (1997) and also Xu *et al.* (1999). Brooks *et al.* (1997) 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. The TS 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.* (in revision).

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 nonused 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 nontabu 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 is usually found that this reduces the sensitivity of the algorithm to selection of the tabu list length. The tabu list length is reset 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 of 14–42 in the first instance and 28–36 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 the 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. Search the neighborhood of all possible defined moves for each subsystem
Note that 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 (Glover and Laguna, 1997).
Moves operate on subsystems only and two kinds are used.

1. 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 ($x_{ij} \to x_{ij}+1$, or $x_{ij} \to x_{ij} - 1$). 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 ($x_{ij} \to x_{ij} + 1$, and $x_{ik} \to x_{ik} - 1$ for $j \neq k$, enumerating all possibilities).
2. For the TSRAP without allowing component mixing, the first type of move is to change the number of components by adding or subtracting one ($x_{ij} \to x_{ij} + 1$, or $x_{ij} \to x_{ij} - 1$), 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 ($x_{ij} \to x_{ik}$ 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. Note that the selected move is not necessarily better than the current solution, it is simply the best available move. 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 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 and Smith (1996b) 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 and Laguna (1997).) Based on numerous findings of its value, a similar approach has been adopted.

An adaptive penalty method has been developed (Kulturel-Konak *et al.*, in revision) 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 the NFT is adaptively moved closer to feasibility as the search progresses. On the other hand, if it is desirable to search deeper into the infeasible region to promote diversity, then the NFT is moved further from the feasible boundary. It is important to note that no user intervention is required to update the NFT as the search progresses. It is done adaptively based on the relative success of the search using predefined 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:

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

$R_{\mathrm{p}}(t_o;\mathbf{x})$ 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. Δc and Δw represent the magnitude of the cost and the weight constraint violations. If a constraint is not violated, then Δc and/or Δw is/are zero. K_1 and K_2 are amplification exponents. The amplification exponents, K_i, are set to two for all work in this paper. However, the method is quite robust to the 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 made 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:

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

$C_{\mathrm{p}}(\mathbf{x})$ 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. Δw and Δr 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, ρ_j, is defined as:

$$\rho_j = \frac{F_j}{T_j}. \tag{4}$$

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 versus infeasible solutions on the tabu list. The method of Gendreau *et al.* (1994) uses a somewhat similar concept in that the penalty changes with recent constraint violations of the last predefined number of solutions. In the method of Gendreau *et al.* (1994) the weight for a constraint that has been violated during the previous h iterations is multiplied by two. The weight for a constraint that has not been violated during the previous 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, a method based on the ρ_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 the NFT for a single constraint. If all moves on the tabu list are infeasible and the current move is also infeasible, then the NFT decreases by a factor of one-half. This increases the penalty for an infeasible solution and moves the search towards the feasible region. This geometric change in the NFT creates a lower bound on the NFT of zero. Alternatively, if all moves on the tabu list are feasible and the current move is also feasible, then it may be beneficial to promote the search into the infeasible region. In this case, the NFT increases by a factor of 1.5. This has the property of encouraging the search towards the infeasible region; thereby promoting diversity in the prospective solution candidates.

If all moves on the tabu list are feasible and the current move is infeasible, then the NFT remains unchanged. Similarly, if all moves on the tabu list are infeasible and the current move is feasible, then the NFT remains unchanged. In these cases, the value of the 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 again, then the 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 $\mathrm{NFT}_{i,j}$ 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 percentage of each constraint as an initial value. For example, the initial values, $\mathrm{NFT}_{r,o}$ and $\mathrm{NFT}_{w,o}$, 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 either discrete or continuous can also 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 used in test problem 3 in Section 4.3. If Problem (P1) is being solved with $C = 400$ and $W = 300$, an initial feasible solution might be:

$$3 \quad 7 \quad 11 \quad 11 \quad 5 \quad 5 \quad 11 \quad 11,$$

which consists of one component of type 3 and one component of type 7 in subsystem 1, and two of component type 5 in subsystem 2 (since there are 10 component choices, entry 11 denotes an empty space, or a blank component). In this case, system cost = 320, system weight = 320 and system reliability = 0.882 459. 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 shown in Table 1. The second kind of move is to replace a component with another type of component. The neighborhood of this move type is shown in Table 2.

Among the solutions of both neighborhoods, the best one is

$$3 \quad 1 \quad 11 \quad 11 \quad 5 \quad 5 \quad 11 \quad 11.$$

Therefore, a move is made to this solution.

Table 1. The neighborhood of the first type of move

*Solution encoding (**x**)*								C(**x**)	W(**x**)	R(**x**)
3	3	7	11	5	5	11	11	400	352	0.948 630
7	11	11	11	5	5	11	11	240	288	0.637 383
3	7	7	11	5	5	11	11	360	418	0.941 832
3	11	11	11	5	5	11	11	280	222	0.710 366
3	7	11	11	5	5	5	11	420	415	0.902 850
3	7	11	11	5	11	11	11	220	225	0.758 127

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.

$$1 \mid 3 \quad 1 \quad 11 \quad 11 \mid 375 \mid 274.$$

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 a GA 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.* (1968) and modified by Nakagawa and Miyazaki (1981). Fyffe *et al.* (1968) 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 and Miyazaki (1981) developed 33 variations of the original problem, where the cost constraint is maintained at 130 and the weight constraint varies from 191–159. The component cost, weight and reliability values were originally presented in Fyffe *et al.* (1968) 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 and Smith (1996a) 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 3. The best solution

Table 2. The neighborhood of the second type of move

*Solution encoding (**x**)*								C(**x**)	W(**x**)	R(**x**)
1	7	11	11	5	5	11	11	335	340	0.966 725
2	7	11	11	5	5	11	11	326	382	0.950 611
4	7	11	11	5	5	11	11	315	380	0.879 102
5	7	11	11	5	5	11	11	301	329	0.875 074
6	7	11	11	5	5	11	11	285	321	0.872 052
7	7	11	11	5	5	11	11	280	386	0.857 280
8	7	11	11	5	5	11	11	276	384	0.846 202
9	7	11	11	5	5	11	11	271	371	0.840 159
10	7	11	11	5	5	11	11	266	354	0.755 557
11	7	11	11	5	5	11	11	240	288	0.637 383
3	**1**	**11**	**11**	**5**	**5**	**11**	**11**	**375**	**274**	**0.968 112**
3	2	11	11	5	5	11	11	366	316	0.955 501
3	3	11	11	5	5	11	11	360	254	0.902 165
3	4	11	11	5	5	11	11	355	314	0.899 537
3	5	11	11	5	5	11	11	341	263	0.896 384
3	6	11	11	5	5	11	11	325	255	0.894 020
3	8	11	11	5	5	11	11	316	318	0.873 789
3	9	11	11	5	5	11	11	311	305	0.869 060
3	10	11	11	5	5	11	11	306	288	0.802 850
3	11	11	11	5	5	11	11	280	222	0.710 366
3	7	11	11	1	5	11	11	357	308	0.896 588
3	7	11	11	2	5	11	11	352	321	0.894 506
3	7	11	11	3	5	11	11	347	319	0.889 747
3	7	11	11	4	5	11	11	342	318	0.885 583
3	7	11	11	6	5	11	11	279	288	0.878 741
3	7	11	11	7	5	11	11	274	290	0.849 145
3	7	11	11	8	5	11	11	261	274	0.822 375
3	7	11	11	9	5	11	11	256	258	0.815 980
3	7	11	11	10	5	11	11	250	276	0.808 544
3	7	11	11	11	5	11	11	220	225	0.758 127
3	7	11	11	5	1	11	11	357	308	0.896 588
3	7	11	11	5	2	11	11	352	321	0.894 506
3	7	11	11	5	3	11	11	347	319	0.889 747
3	7	11	11	5	4	11	11	342	318	0.885 583
3	7	11	11	5	6	11	11	279	288	0.878 741
3	7	11	11	5	7	11	11	274	290	0.849 145
3	7	11	11	5	8	11	11	261	274	0.822 375
3	7	11	11	5	9	11	11	256	258	0.815 980
3	7	11	11	5	10	11	11	250	276	0.808 544
3	7	11	11	5	11	11	11	220	225	0.758 127

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 (Coit and Smith, 1996a) 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 3 indicate, TSRAP generally outperformed the GA. TSRAP obtained a higher reliability in 22 of the 33 test cases. Additionally, it had a lower standard deviation in 21 of the 33 test cases. Thus, TSRAP generally yielded solutions with a higher reliability and was more

Table 3. Test problem 1: a comparison of the GA of Coit and Smith (1996a) and the TSRAP

			GA (Coit and Smith, 1996a)—10 runs		*TSRAP—10 runs*			
Case	C	W	*Max* R	*Std dev*	*Max* R	*Std dev*	*Average evaluation*	*%MPI*
1	130	191	0.986 70	0.000 422	0.986 811	0.000 644	347 969	0.83
2	130	190	0.985 70	0.000 595	0.986 416	0.000 000	503 369	5.01
3	130	189	0.985 60	0.000 853	0.985 922	0.000 000	484 283	2.24
4	130	188	0.985 00	0.000 306	0.985 378	0.000 765	387 029	2.52
5	130	187	0.984 40	0.000 748	0.984 688	0.000 307	347 116	1.85
6	130	186	0.983 60	0.000 586	0.984 176	0.000 386	413 235	3.51
7	130	185	0.983 10	0.000 856	0.983 505	0.000 510	366 652	2.40
8	130	184	0.982 30	0.000 465	0.982 994	0.000 450	367 142	3.92
9	130	183	0.981 90	0.000 389	0.982 256	0.000 628	361 935	1.97
10	130	182	0.981 10	0.000 381	0.981 518	0.000 428	402 144	2.21
11	130	181	0.980 20	0.000 842	0.981 027	0.000 356	367 933	4.18
12	130	180	0.979 70	0.000 791	0.980 290	0.000 469	436 098	2.91
13	130	179	0.979 10	0.000 538	0.979 505	0.000 341	303 575	1.94
14	130	178	0.978 30	0.000 701	0.978 400	0.000 143	262 093	0.46
15	130	177	0.977 20	0.001 031	0.977 474	0.000 198	344 122	1.20
16	130	176	0.976 40	0.000 751	0.976 690	0.000 669	301 944	1.23
17	130	175	0.975 30	0.000 795	0.975 708	0.000 288	345 226	1.65
18	130	174	0.974 35	0.000 812	0.974 788	0.000 610	276 510	1.71
19	130	173	0.973 62	0.000 753	0.973 827	0.000 534	260 570	0.78
20	130	172	0.972 66	0.001 083	0.973 027	0.000 166	355 909	1.34
21	130	171	0.971 86	0.000 812	0.971 929	0.000 000	334 564	0.25
22	130	170	0.970 76	0.000 821	0.970 760	0.000 490	311 927	0.00
23	130	169	0.969 22	0.000 415	0.969 291	0.000 351	310 691	0.23
24	130	168	0.968 13	0.000 596	0.968 125	0.000 957	320 568	−0.02
25	130	167	0.966 34	0.000 304	0.966 335	0.000 739	395 987	−0.01
26	130	166	0.965 04	0.000 569	0.965 042	0.000 616	322 723	0.01
27	130	165	0.963 71	0.000 474	0.963 712	0.000 932	315 828	0.01
28	130	164	0.962 42	0.000 659	0.962 422	0.001 044	413 602	0.01
29	130	163	0.960 64	0.000 401	0.959 980	0.000 176	347 964	−1.68
30	130	162	0.959 12	0.000 833	0.958 205	0.000 053	305 090	−2.24
31	130	161	0.958 03	0.000 808	0.956 922	0.001 230	256 383	−2.64
32	130	160	0.955 67	0.000 473	0.955 604	0.001 424	343 186	−0.15
33	130	159	0.954 32	0.000 363	0.954 325	0.002 004	346 894	0.01

%MPI = 100% × (TSRAP max − GA max)/(1 − GA max).

consistent. Since the GA contains many more stochastic elements, it is not surprising that the TSRAP is more consistent across the 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 1200 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 3 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 recalculation 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 the computation time by 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 and Liu (2000) 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 and Miyazaki (1981). For each of 14 subsystems, component choices with cost, weight and exponential distribution parameter (λ_{ij}) are given in Table 4. 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 is 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 the 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 and Liu (2000) 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 5. 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 either not known or it 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 8 hours might be selected as a typical overseas flight. (Neither Fyffe *et al.* (1968) nor Nakagawa and Miyazaki (1981) stated a mission time.) If each aircraft makes six overseas flights per week, an increase in system reliability of 0.003 08, as achieved in the last instance, would translate to an expected 58 fewer failures over the course of a year ($60 \times 6 \times 52 \times 0.003\,08$). Also, when system failure is of high consequence, as in a nuclear power plant for instance, then 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 and Smith (1996a). The component choices are shown in Table 6. This sample problem was designed with two subsystems where $k_1 = 4$ and $k_2 = 2$ (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 10

Table 4. Component data for test problem 2

Sub system		*Component choices*											
		1			*2*			*3*			*4*		
i	k_i	λ^*_{1j}	c_{1j}	w_{1j}	λ_{2j}	c_{2j}	w_{2j}	λ_{3j}	c_{3j}	w_{3j}	λ_{4j}	c_{4j}	w_{4j}
1	1	0.001 054	1	3	0.000 726	1	4	0.000 943	2	2	0.000 513	2	5
2	2	0.000 513	2	8	0.000 619	1	10	0.000 726	1	9	—		
3	1	0.001 625	2	7	0.001 054	3	5	0.001 393	1	6	0.000 834	4	4
4	2	0.001 863	3	5	0.001 393	4	6	0.001 625	5	4	—		
5	1	0.000 619	2	4	0.000 726	2	3	0.000 513	3	5	—		
6	2	0.000 101	3	5	0.000 202	3	4	0.000 305	2	5	0.000 408	2	4
7	1	0.000 943	4	7	0.000 834	4	8	0.000 619	5	9	—		
8	2	0.002 107	3	4	0.001 054	5	7	0.000 943	6	6	—		
9	3	0.000 305	2	8	0.000 101	3	9	0.000 408	4	7	0.000 943	3	8
10	3	0.001 863	4	6	0.001 625	4	5	0.001 054	5	6	—		
11	3	0.000 619	3	5	0.000 513	4	6	0.000 408	5	6	—		
12	1	0.002 357	2	4	0.001 985	3	5	0.001 625	4	6	0.001 054	5	7
13	2	0.000 202	2	5	0.000 101	3	5	0.000 305	2	6	—		
14	3	0.001 054	4	6	0.000 834	4	7	0.000 513	5	6	0.000 101	6	9

*Units for λ_{ij} are failures/hour.

Table 5. Test problem 2: a comparison of IP and TSRAP

			IP	*TSRAP (no mixing)—10 runs*			*TSRAP (mixing)—10 runs*			
Case	C	W	R	*Max* R	*Std dev*	*Average evaluations*	*Max* R	*Std dev*	*Average evaluations*	*%MPI*
1	130	191	0.606 65	0.606 65	0.000 000	83 876	0.607 66	0.000 000	343 333	0.26
2	130	190	0.590 64	0.590 64	0.000 583	123 766	0.595 38	0.000 610	377 963	1.16
3	130	189	0.581 05	0.581 05	0.000 345	91 708	0.588 27	0.000 000	368 893	1.72
4	130	188	0.573 27	0.573 27	0.001 371	96 616	0.577 34	0.001 508	289 590	0.95
5	130	187	0.562 50	0.562 50	0.000 491	106 152	0.570 44	0.002 255	265 562	1.81
6	130	186	0.554 98	0.554 98	0.000 000	90 123	0.558 92	0.000 910	292 436	0.88
7	130	185	0.542 85	0.542 85	0.000 698	105 121	0.549 36	0.000 523	279 319	1.42
8	130	184	0.534 01	0.534 01	0.000 238	83 484	0.541 12	0.001 718	287 372	1.53
9	130	183	0.529 81	0.529 81	0.000 000	104 231	0.532 26	0.000 194	367 116	0.52
10	130	182	0.522 72	0.522 72	0.000 000	95 722	0.523 85	0.002 439	282 638	0.24
11	130	181	0.512 90	0.512 90	0.000 549	86 564	0.515 28	0.002 924	311 123	0.49
12	130	180	0.506 04	0.506 04	0.000 000	106 757	0.506 04	0.002 000	232 905	**0.00**
13	130	179	0.490 57	0.490 57	0.001 899	85 898	0.494 02	0.002 297	263 555	0.68
14	130	178	0.484 68	0.484 68	0.000 000	90 370	0.485 73	0.001 836	291 947	0.20
15	130	177	0.474 91	0.474 91	0.000 277	109 807	0.477 78	0.004 778	290 110	0.55
16	130	176	0.469 22	0.469 22	0.000 000	85 526	0.469 22	0.005 300	258 560	**0.00**
17	130	175	0.452 98	0.452 98	0.000 000	94 737	0.457 43	0.000 327	222 826	0.81
18	130	174	0.448 78	0.448 78	0.001 711	116 496	0.449 75	0.006 295	243 874	0.18
19	130	173	0.439 23	0.439 23	0.000 226	97 602	0.442 39	0.005 074	312 906	0.56
20	130	172	0.434 46	0.434 46	0.000 000	98 716	0.434 46	0.005 173	266 489	**0.00**
21	130	171	0.419 42	0.419 42	0.001 902	97 759	0.420 33	0.003 418	255 287	0.16
22	130	170	0.410 50	0.410 50	0.000 030	107 748	0.413 45	0.002 625	245 476	0.50
23	130	169	0.406 04	0.406 04	0.003 841	72 328	0.406 04	0.004 025	220 822	**0.00**
24	130	168	0.390 25	0.390 25	0.000 000	88 613	0.391 09	0.000 000	287 080	0.14
25	130	167	0.381 94	0.381 94	0.000 000	80 788	0.384 69	0.000 000	244 641	0.44
26	130	166	0.377 79	0.377 79	0.000 000	117 293	0.377 79	0.000 000	251 777	**0.00**
27	130	165	0.364 72	0.364 72	0.000 000	83 956	0.365 50	0.000 729	263 279	0.12
28	130	164	0.356 96	0.356 96	0.001 869	85 885	0.359 52	0.000 000	309 496	0.40
29	130	163	0.353 08	0.353 08	0.000 000	100 222	0.353 08	0.000 000	241 806	**0.00**
30	130	162	0.332 43	0.332 43	0.000 000	103 432	0.332 43	0.001 725	243 964	**0.00**
31	130	161	0.323 92	0.323 92	0.000 000	97 440	0.323 92	0.002 312	275 075	**0.00**
32	130	160	0.309 08	0.309 08	0.001 085	96 723	0.312 09	0.000 000	309 487	0.43
33	130	159	0.302 50	0.302 50	0.009 224	75 507	0.305 58	0.000 000	230 889	0.44

%MPI = 100% × (TSRAP max rel. − IP rel.)/(1 − IP rel.).

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 7. Twenty runs of TSRAP were performed and compared with the previous GA results (Coit and Smith, 1996a) in Table 7 along with the optimal solutions found by enumeration shown in Table 8. 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.

Table 6. 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

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

Table 7. Comparison of the GA and TSRAP for test problem 3

Problem description					*GA (Coit and Smith, 1996a)—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 the Nakagawa and Miyazaki (1981) and also Bulfin and Liu (1985) formulations.

Table 8. 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

infeasible region with an adaptively changing NFT. TSRAP was demonstrated on three test problems with encouraging results. Like GAs, the TS methodology can be applied to many diverse reliability optimization problems where mathematical programming approaches have not been successful. When compared to GAs, TSRAP results in a 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.

Acknowledgment

This research was conducted while Sadan Kulturel-Konak was a Ph.D. candidate in the Industrial and Systems Engineering Department at Auburn University, AL. She would like to thank Dr. Abdullah Konak for valuable discussions during the research. David W. Coit's contributions were supported by NSF CAREER grant DMII-987416.

References

Bellman, R.E. (1957) *Dynamic Programming*, Princeton University Press, Princeton, NJ.

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

Bellman, R.E. and Dreyfus, E. (1962) *Applied Dynamic Programming*, Princeton University Press, Princeton, NJ.

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

Bland, J.A. (1998b) Structural design optimization with reliability constraints using tabu search. *Engineering Optimization*, **30**(1), 55–74.

Brooks, R.R., Iyengar, S.S. and Rai, S. (1997) Minimizing cost of redundant sensor-systems with non-monotone and monotone search algorithms, in *Proceedings of the Annual Reliability and Maintainability Symposium*, IEEE, New York, pp. 307–313.

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

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

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

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

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

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

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

Gen, M., Ida, K., Tsujimura, Y. and Kim, C.E. (1993) Large scale 0-1 fuzzy goal programming and its application to reliability optimization problem. *Computers & Industrial Engineering*, **24**, 539–549.

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

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

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

Glover, F. (1989) Tabu search-part I. *ORSA Journal of Computing*, **1**, 190–206.

Glover, F. (1990) Tabu search-part II. *ORSA Journal of Computing*, **2**, 4–32.

Glover, F. and Laguna, M. (1997) *Tabu Search*, Kluwer, London, UK.

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

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

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

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

Liang, Y.C. and Smith, A.E. (1999) An ant system approach to redundancy allocation, in *Proceedings of the 1999 Congress on Evolutionary Computation*, IEEE, Piscataway, NJ, pp. 1478–1484.

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

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

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

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

Reeves, C.R. (1993) *Modern Heuristic Techniques for Combinatorial Problems*, Wiley, New York, NY.

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

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

Xu, J., Chiu, S.Y. and Glover, F. (1999) Optimizing a ring-based private line telecommunication network using tabu search. *Management Science*, **45**(3), 330–345.

Biographies

Sadan Kulturel-Konak is currently an Assistant Professor of Management Information Systems at Penn Berks-Lehigh Valley College. She received her degrees in Industrial Engineering: B.S. from Gazi University, Turkey in 1993, M.S. from the Middle East Technical University, Turkey in 1996 and from the University of Pittsburgh in 1999, and Ph.D. from Auburn University in 2002. Her research interests are in modeling and optimization of complex systems and robustness under uncertainty with applications to facility layout. She is a member of INFORMS, IIE and the Operations Research Society of Turkey.

Alice E. Smith is Philpott-WestPoint Stevens Professor and Chair of Industrial and Systems at Auburn University. Previous to this position, she was on the faculty of the Department of Industrial Engineering at the University of Pittsburgh, which she joined in 1991 after 10 years of industrial experience with Southwestern Bell Corporation. She has degrees in engineering and business from Rice University, Saint Louis University and University of Missouri—Rolla. Her research has been funded by the National Institute of Standards (NIST), Lockheed Martin, Adtranz NA, the Ben Franklin Technology Center of Western Pennsylvania, Daimler-Chrysler, and the National Science Foundation (NSF), from which she was awarded a CAREER grant in 1995 and an ADVANCE Leadership grant in 2001. She served in an editorial capacity for *IIE Transactions, INFORMS Journal on Computing* and *IEEE Transactions on Evolutionary Computation* and has authored over 100 refereed publications. She is a senior member of IIE, IEEE and SWE, a member of Tau Beta Pi, INFORMS and ASEE, and a Registered Professional Engineer in Industrial Engineering in Alabama and Pennsylvania. She currently serves as Senior Vice President (Academic) of IIE.

David W. Coit is an Associate Professor in the Industrial and Systems Engineering Department at Rutgers University. He received a B.S. degree in Mechanical Engineering from Cornell University, an M.B.A. from Rensselaer Polytechnic Institute and M.S. and Ph.D. in Industrial Engineering from the University of Pittsburgh. He also has over 10 years of experience working for the IIT Research Institute (IITRI), Rome NY where he was a reliability analyst and project manager, and in his final position, the Manager of Engineering at IITRI's Assurance Technology Center. In 1999, he was awarded a CAREER grant from NSF. His current research involves reliability prediction and optimization, and multi-criteria optimization considering uncertainty. He is a member of IIE, IEEE and INFORMS.

Contributed by the Reliability Engineering Department