

An Ant Colony Optimization Algorithm for the Redundancy Allocation Problem (RAP)

Yun-Chia Liang, Associate Member, IEEE and Alice E. Smith, Senior Member, IEEE

Abstract—This paper uses an ant colony meta-heuristic optimization method to solve the redundancy allocation problem (RAP). The RAP is a well known NP-hard problem which has been the subject of much prior work, generally in a restricted form where each subsystem must consist of identical components in parallel to make computations tractable. Meta-heuristic methods overcome this limitation, and offer a practical way to solve large instances of the relaxed RAP where different components can be placed in parallel. The ant colony method has not yet been used in reliability design, yet it is a method that is expressly designed for combinatorial problems with a neighborhood structure, as in the case of the RAP. An ant colony optimization algorithm for the RAP is devised & tested on a well-known suite of problems from the literature. It is shown that the ant colony method performs with little variability over problem instance or random number seed. It is competitive with the best-known heuristics for redundancy allocation.

Index Terms—Ant colony optimization, combinatorial optimization, genetic algorithm, redundancy allocation, series-parallel system.

Abbreviations & Acronyms

ACO	Ant Colony Optimization
GA	Genetic Algorithm
RAP	Redundancy Allocation Problem

Notation

RAP

k	minimum number of components required to function a pure parallel system
n	total number of components used in a pure parallel system
k - <i>out-</i> <i>of-n:</i>	a system that functions when at least k of its n components function
G	overall reliability of the series-parallel system
C	cost constraint
W	weight constraint
s	number of subsystems
a_i	number of available component choices for subsystem i
r_{ij}	reliability of component j available for subsystem i

c_{ij}	cost of component j available for subsystem i
w_{ij}	weight of component j available for subsystem i
y_{ij}	quantity of component j used in subsystem i
Y_i	$(y_{i1}, \dots, y_{ia_i})$
n_i	$= \sum_{j=1}^{a_i} y_{ij}$, total number of components used in subsystem i
n_{\max}	maximum number of components which can be in parallel (user assigned)
k_i	minimum number of components in parallel required for subsystem i to function
$R_i(y_i k_i)$	reliability of subsystem i , given k_i
$C_i(y_i)$	total cost of subsystem i
$W_i(y_i)$	total weight of subsystem i
R_u	unpenalized system reliability of solution u
R_{up}	penalized system reliability of solution u
R_{mp}	penalized system reliability of the rank m^{th} solution
C_u	total system cost of solution u
W_u	total system weight of solution u
AC	set of available component choices

ACO

i	index for subsystem, $i = 1, \dots, s$
j	index for components in a subsystem
τ_{ij}	pheromone trail intensity of combination (i, j)
τ_{ij}^{old}	pheromone trail intensity of combination (i, j) before update
τ_{ij}^{new}	pheromone trail intensity of combination (i, j) after update
τ_{i0}	$= 1/a_i$, initial pheromone trail intensity of subsystem i
P_{ij}	transition probability of combination (i, j)
η_{ij}	problem-specific heuristic of combination (i, j)
α	relative importance of the pheromone trail intensity
β	relative importance of the problem-specific heuristic
l	index for component choices from set AC
ρ	$\in [0, 1]$, trail persistence
q	$\in [0, 1]$, a uniformly generated random number
q_0	$\in [0, 1]$, a parameter which determines the relative importance of exploitation versus exploration
E	number of best solutions chosen for offline pheromone update
m	index (rank, best to worst) for solutions in a given iteration
γ	amplification parameter in the penalty function

I. INTRODUCTION

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 i is functioning properly if at least k_i

Manuscript received December 1, 2001; revised April 1, 2003. Associate Editor: J. H. Lambert.

Y.-C. Liang is with the Department of Industrial Engineering and Management, Yuan Ze University, Chungli, Taoyuan 320, Taiwan, R.O.C. (e-mail: ycliang@saturn.yzu.edu.tw).

A. E. Smith is with the Department of Industrial and Systems Engineering, Auburn University, Auburn, AL 36849 USA (e-mail: a-smith@eng.auburn.edu).

Digital Object Identifier 10.1109/TR.2004.832816

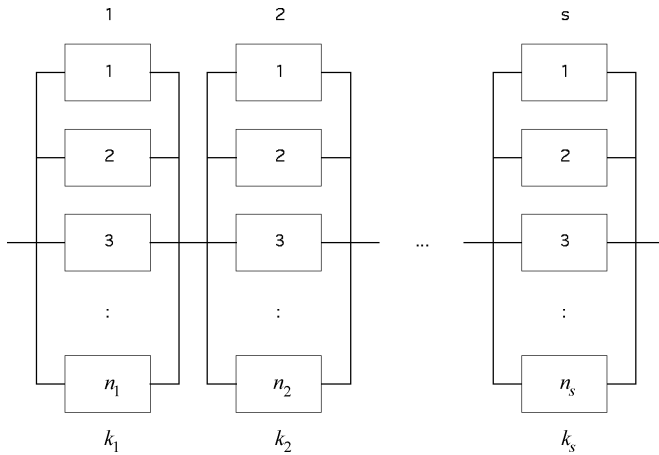


Fig. 1. Series-parallel system configuration.

of its n_i components are operational. A series-parallel system is where $k_i = 1$ for all subsystems. In the formulation of a series-parallel system problem, for each subsystem, multiple component choices are used in parallel. Thus, the problem is to select the optimal combination of components and redundancy levels to meet system level constraints while maximizing system reliability. The RAP is NP-hard [6], and has been studied in many forms as summarized in Tillman, *et al.* [31], and more recently by Kuo & Prasad [23].

The RAP can be formulated to maximize system reliability given restrictions on system cost of C and system weight of W . It is assumed that system weight and system cost are linear combinations of component weight and cost, although this is a restriction which can often be relaxed using heuristics.

$$\max R = \prod_{i=1}^s R_i(y_i | k_i) \quad (1)$$

Subject to the constraints

$$\sum_{i=1}^s C_i(y_i) \leq C,$$

$$\sum_{i=1}^s W_i(y_i) \leq W,$$

If there is a pre-selected maximum number of components allowed in parallel, the following constraint is added:

$$k_i \leq \sum_{j=1}^{a_i} y_{ij} \leq n_{\max} \quad \forall i = 1, 2, \dots, s$$

Typical assumptions are:

- The states of components and the system are either good or failed.
- Failed components do not damage the system, and are not repaired.
- The failure rates of components when not in use are the same as when in use (i.e., active redundancy).
- Component attributes (reliability, weight, and cost) are known and deterministic.
- The supply of components is unlimited.

Traditional exact optimization approaches to the RAP include dynamic programming (e.g., [2], [19], [27]), integer programming (e.g., [3], [21], [22], [26]), and mixed-integer & nonlinear programming (e.g., [32]). Because of the exponential increase in

search space with problem size, heuristics have become a popular alternative to exact methods. Meta-heuristics, in particular, offer flexibility while not being confined to specific problem types or instances (hence, the “meta”). GA meta-heuristics have been proposed by Painton & Campbell [28], Levitin *et al.* [24], and Coit & Smith [7], [8]. For a fixed design configuration, and known incremental decreases in component failure rates & their associated costs, [28] uses a GA to find the maximum 5th percentile of the mean-time-between-failure distribution reliability design to satisfy specific cost constraints. Levitin *et al.* [24] generalize a redundancy allocation problem to multi-state systems, where the system and its components have a range of performance levels—from perfectly functioning to complete failure. A universal moment generating function is used to estimate system performance (capacity or operation time), and a GA is employed as the optimization technique. Coit & Smith [7] use a GA which searches over feasible and infeasible regions to identify a final, feasible optimal, or near optimal solution to a relaxed version of the RAP. Coit & Smith [8] also applied GA to problems with stochastic system reliability, and mean-time-to-failure.

Because of the large search space size of the RAP, and the lack of a solution technique that dominates others, it is a good candidate for other meta-heuristic approaches including the focus of this paper, ACO.

II. THE ACO APPROACH

ACO is one of the adaptive meta-heuristic optimization methods inspired by nature which include simulated annealing, GA, and tabu search. ACO is distinctly different from other meta-heuristic methods in that it *constructs* an entire new solution set (colony) in each generation, while others focus on *improving* the set of solutions or a single solution from previous iterations. ACO was inspired by the behavior of real ants. Ethologists have studied how blind animals, such as ants, could establish shortest paths from their nest to food sources. The medium that is used to communicate information among individual ants regarding paths is pheromone. A moving ant lays some pheromone on the ground, thus marking the path. The pheromone, while gradually dissipating over time, is reinforced as other ants use the same trail. Therefore, efficient trails increase their pheromone level over time while poor ones reduce to nil. Inspired by the behavior of real ants, Marco Dorigo introduced the ant colony optimization approach in his Ph.D. thesis in 1992 [13], and expanded it in his further work, as summarized in [14], [15], [18]. The characteristics of ant colony optimization include:

- 1) a method to construct solutions which balances pheromone trails (characteristics of past solutions) with a problem-specific heuristic (normally, a simple greedy rule),
- 2) a method to both reinforce & evaporate pheromone, and
- 3) local (neighborhood) search to improve solutions.

ACO methods have been successfully applied to diverse combinatorial optimization problems including traveling salesman [16], [17], quadratic assignment [25], [30], vehicle routing [4], [5], [20], telecommunication networks [12], graph coloring [10], constraint satisfaction [29], Hamiltonian graphs [33], and scheduling [1], [9], [11].

A. Solution Encoding

Each ant represents a design of the entire system, a collection of n_i components in parallel ($k_i \leq n_i \leq n_{\max}$) for s different subsystems. The n_i components can be chosen in any combination from the a_i available types of components. The a_i types are indexed in descending order of reliability; i.e., 1 represents the most reliable component type, etc. An index of $a_i + 1$ is assigned to a position where an additional component was not used (left blank), i.e., with attributes of zero. Each of the s subsystems is represented by n_{\max} positions with each component listed according to its reliability index, as in [7]. Therefore, a complete solution (ant) consists of an integer vector of length $n_{\max} \times s$.

B. Solution Construction

In the ACO-RAP algorithm, ants use problem-specific heuristic information, denoted by η_{ij} , as well as pheromone trail intensity, denoted by τ_{ij} , to construct a solution. n_i components ($k_i + 1 \leq n_i \leq n_{\max} - 4$) are selected for each subsystem using the probabilities established by (2) & (3), below. The selection of this range of components is to encourage the construction of a solution that is likely to be feasible, that is, be reliable enough (the $k_i + 1$ lower bound) but not violate the weight and cost constraints (the $n_{\max} - 4$ upper bound). Solutions which contain more or less components per subsystem than these bounds are examined during the local search phase of the algorithm (described in Section II-D).

The metric chosen for the ACO problem specific heuristic is $\eta_{ij} = r_{ij}/(c_{ij} + w_{ij})$, where r_{ij} , c_{ij} , and w_{ij} represent the associated reliability, cost, and weight of component j for subsystem i . Components with higher reliability, and smaller cost and weight have greater probability of selection. In keeping with the ACO meta-heuristic concept, this is a simple and obvious rule. The pheromone trail intensities for the initial iteration (colony of ants) are uniformly set over the component choices, that is, $\tau_{i0} = 1/a_i$. The pheromone trail intensities are subsequently modified as described in Section II-E.

A solution is constructed by selecting component j for subsystem i according to:

$$j = \begin{cases} \arg \max_{l \in AC} [(\tau_{il})^\alpha (\eta_{il})^\beta] & q \leq q_0 \\ J & q > q_0 \end{cases}, \quad (2)$$

and J is selected according to the transition probability mass function given by

$$P_{ij} = \begin{cases} \frac{(\tau_{ij})^\alpha (\eta_{ij})^\beta}{\sum_{l=1}^{a_i} (\tau_{il})^\alpha (\eta_{il})^\beta} & j \in AC \\ 0 & \text{Otherwise} \end{cases}; \quad (3)$$

where α , β and are parameters which control the relative weight of the pheromone, and the local heuristic, respectively; AC is the set of available component choices for subsystem i ; q is a random number uniformly generated between 0 & 1; and q_0 is a parameter which determines the relative importance of the exploitation of good solutions versus the exploration of search spaces. When $q \leq q_0$, exploitation of known good solutions occurs. The component selected is the best for that particular subsystem; that is, it has the highest product of pheromone intensity, and ratio of reliability to cost & weight. When $q > q_0$,

the search favors more exploration as all components are considered for selection with some probability.

C. Objective Function

After all components of solution u are selected, the unpenalized reliability R_u is calculated using (1). For solutions with costs that exceed C and/or weight that exceeds W , a penalized reliability R_{up} is calculated:

$$R_{up} = R_u \cdot \left(\frac{W}{W_u}\right)^\gamma \cdot \left(\frac{C}{C_u}\right)^\gamma \quad (4)$$

where the exponent γ is a preset amplification parameter and W_u and C_u represent the total system weight and cost of solution u , respectively. This penalty function encourages the ACO-RAP algorithm to explore the feasible region and infeasible region that is near the border of the feasible area, and discourages, but permits, search further into the infeasible region.

D. Improving Constructed Solutions Through Local Search

After a colony is generated, each solution is improved using local search. Starting with the first subsystem, a chosen component type is deleted, and another component type is added. All possibilities are enumerated. For example, if a subsystem has one of component 1, two of component 2, and one of component 3; then one alternative is to delete a component 1, and to add a component 2. Another possibility is to delete a component 3, and to add a component 1. Whenever an improvement of the objective function is detected, the new solution replaces the old one, and the process continues until all subsystems have been searched. This local search method does not require recalculating the entire system reliability each time. Only the reliability of the subsystem under consideration needs to be recalculated, and system reliability is updated accordingly.

E. Pheromone Trail Intensity Update

In ACO, the pheromone trail intensity update consists of two phases: i) online (ant-by-ant) updating, and ii) offline (colony) updating. Online updating is done after each solution is constructed, and its purpose is to decay the pheromone intensity of the components of the solution just constructed to encourage exploration of other component choices in the subsequent solutions to be constructed. Online updating is by

$$\tau_{ij}^{new} = \rho \cdot \tau_{ij}^{old} + (1 - \rho) \cdot \tau_{i0} \quad (5)$$

where $\rho \in [0, 1]$ is a parameter that controls the pheromone persistence; i.e., $1 - \rho$ represents the proportion of the pheromone evaporated. After all solutions in a colony (an iteration) have been constructed and improved by local search, pheromone trail intensities are updated offline. Offline updating is to reflect the discoveries of this iteration. The offline intensity update is:

$$\tau_{ij}^{new} = \rho \cdot \tau_{ij}^{old} + (1 - \rho) \cdot \sum_{m=1}^E (E - m + 1) \cdot R_{mp}, \quad (6)$$

where $m = 1$ is the best feasible solution yet found (which may or may not be in the current colony), and the remaining $E - 1$ solutions are the best ones in the current colony. In other words, only the best E ants are allowed to contribute their pheromone

to the trail intensity, and the magnitudes of contributions are weighted by their ranks in the colony.

The flow of the algorithm can be expressed as follows:

Set all parameters, and initialize the pheromone trail intensities.

Loop

 Sub-Loop

 Construct an ant using the pheromone trail intensity, and the problem-

 specific heuristic (Equations 2 & 3).

 Apply the online pheromone intensity update rule (Equation 5).

 Continue until all ants in the colony have been generated.

 Apply local search to each ant in the colony.

 Evaluate all ants in the colony (Equations 1 & 4), rank them, and record the best

 feasible one.

 Apply the offline pheromone intensity update rule (Equation 6).

 Continue until a stopping criterion is reached.

III. COMPUTATIONAL EXPERIENCE

The ACO is coded in Borland C++, and run using an Intel Pentium III 800 MHz PC with 256 MB RAM. All computations use real float point precision without rounding or truncating values. The system reliability of the final solution is rounded to four digits behind the decimal point in order to compare with results in the literature.

The parameters of the ACO algorithm are set to the following values: $\alpha = 1$, $\beta = 0.5$, $q_0 = 0.9$, $\rho = 0.9$, and $E = 5$. This gives relatively more weight to the pheromone trail intensity than the problem-specific heuristic, and greater emphasis on exploitation rather than exploration. The ACO is not very sensitive to changes in these values, and tested well for quite a range of them. For the penalty function, $\gamma = 0.1$ except when the previous iteration has 90% or more infeasible solutions, then $\gamma = 0.3$. This increases the penalty temporarily to move the search back into the feasible region when all or nearly all solutions in the current iteration are infeasible. It was found that this dual level penalty improved performance on the most constrained instances of the test problems. Because of varying magnitudes of R , C , and W ; all η_{ij} & τ_{ij} are normalized between (0,1) before solution construction ((2) & (3)). 100 ants are used in each iteration. The stopping criterion is when the number of iterations reaches 1000, or the best feasible solution has not changed for 500 consecutive iterations. This results in a maximum of 100 000 ants.

The 33 variations of the Fyffe *et al.* problem [19], as devised by Nakagawa & Miyazaki [27], were used to test the performance of ACO. In this problem set, $C = 130$, and W is decreased incrementally from 191 to 159. In [19] & [27], the op-

TABLE I
COMPARISON OF THE GA [7], AND THE ACO OVER 10 RANDOM NUMBER SEEDS
EACH FOR THE TEST PROBLEMS FROM [27]

No	C	W	C&S [7] GA - 10 runs			ACO-RAP - 10 runs		
			Max R	Mean R	Min R	Max R	Mean R	Min R
1	130	191	0.9867	0.9862	0.9854	0.9868	0.9862	0.9860
2	130	190	0.9857	0.9855	0.9852	0.9859	0.9858	0.9857
3	130	189	0.9856	0.9850	0.9838	0.9858	0.9853	0.9852
4	130	188	0.9850	0.9848	0.9842	0.9853	0.9849	0.9848
5	130	187	0.9844	0.9841	0.9835	0.9847	0.9841	0.9837
6	130	186	0.9836	0.9833	0.9827	0.9838	0.9836	0.9835
7	130	185	0.9831	0.9826	0.9822	0.9835	0.9830	0.9828
8	130	184	0.9823	0.9819	0.9812	0.9830	0.9824	0.9820
9	130	183	0.9819	0.9814	0.9812	0.9822	0.9818	0.9817
10	130	182	0.9811	0.9806	0.9803	0.9815	0.9812	0.9806
11	130	181	0.9802	0.9801	0.9800	0.9807	0.9806	0.9804
12	130	180	0.9797	0.9793	0.9782	0.9803	0.9798	0.9796
13	130	179	0.9791	0.9786	0.9780	0.9795	0.9795	0.9795
14	130	178	0.9783	0.9780	0.9764	0.9784	0.9784	0.9783
15	130	177	0.9772	0.9771	0.9770	0.9776	0.9776	0.9776
16	130	176	0.9764	0.9760	0.9751	0.9765	0.9765	0.9765
17	130	175	0.9753	0.9753	0.9753	0.9757	0.9754	0.9753
18	130	174	0.9744	0.9732	0.9716	0.9749	0.9747	0.9741
19	130	173	0.9738	0.9732	0.9719	0.9738	0.9735	0.9731
20	130	172	0.9727	0.9725	0.9712	0.9730	0.9726	0.9714
21	130	171	0.9719	0.9712	0.9701	0.9719	0.9717	0.9710
22	130	170	0.9708	0.9705	0.9695	0.9708	0.9708	0.9708
23	130	169	0.9692	0.9689	0.9684	0.9693	0.9693	0.9693
24	130	168	0.9681	0.9674	0.9662	0.9681	0.9681	0.9681
25	130	167	0.9663	0.9661	0.9657	0.9663	0.9663	0.9663
26	130	166	0.9650	0.9647	0.9636	0.9650	0.9650	0.9650
27	130	165	0.9637	0.9632	0.9627	0.9637	0.9637	0.9637
28	130	164	0.9624	0.9620	0.9609	0.9624	0.9624	0.9624
29	130	163	0.9606	0.9602	0.9592	0.9606	0.9606	0.9606
30	130	162	0.9591	0.9587	0.9579	0.9592	0.9592	0.9592
31	130	161	0.9580	0.9572	0.9561	0.9580	0.9580	0.9580
32	130	160	0.9557	0.9556	0.9554	0.9557	0.9557	0.9557
33	130	159	0.9546	0.9538	0.9531	0.9546	0.9546	0.9546

timization approaches required that only identical components be placed in redundancy; however, for the ACO approach, as in Coit & Smith [7], different types are allowed to reside in parallel (assuming that a value of $n_{\max} = 8$ for all subsystems). Allowing component mixing within a subsystem, the search space size is larger than 7.6×10^{33} . Because the heuristic benchmark for the RAP where component mixing is allowed is the GA of [7], it is chosen for comparison. Ten runs of each algorithm were made using different random number seeds for each problem instance.

The results are summarized in Table I where the comparisons between the GA and ACO results over 10 runs are divided into three categories: maximum, mean, and minimum system reliability (denoted by Max R, Mean R and Min R, respectively). The ACO solutions are equivalent to or superior to the GA in all categories and all problem instances. When the problem instances are less constrained (the first 18), the ACO performs much better than the GA. When the problems become more constrained (the last 15), ACO is equal to GA for 12 instances and better than GA for 3 instances in terms of the Max R measure (best over 10 runs). However, for Min R (worst over 10 runs), and Mean R (of 10 runs), ACO dominates GA. Thus, the ACO tends to find better solutions than the GA, is significantly less sensitive to random number seed, and for the 12 most constrained instances,

TABLE II
CONFIGURATION, RELIABILITY, COST, AND WEIGHT OF THE BEST SOLUTION TO EACH PROBLEM

No.	W	R	Cost	Weight	Solution
1	191	0.9868	130	191	333,11,111,2222,333,22,333,3333,23,122,333,4444,12,12
2	190	0.9859	129	190	333,11,111,2222,333,22,333,3333,22,112,333,4444,11,22
3	189	0.9858	130	189	333,11,111,2222,333,22,333,3333,22,122,11,4444,11,12
4	188	0.9853	130	188	333,11,111,2222,333,22,333,3333,23,112,13,4444,12,12
5	187	0.9847	130	187	333,11,111,2222,333,22,333,3333,23,122,13,4444,11,12
6	186	0.9838	129	186	333,11,111,2222,333,22,333,3333,22,122,11,4444,11,22
			130	186	333,11,111,2222,333,24,333,3333,33,122,13,4444,12,12
7	185	0.9835	130	185	333,11,111,2222,333,22,333,3333,13,122,13,4444,11,22
8	184	0.9830	130	184	333,11,111,222,333,22,333,3333,33,112,11,4444,11,12
9	183	0.9822	128	183	333,11,111,222,333,22,333,3333,33,112,13,4444,11,12
10	182	0.9815	127	182	333,11,111,222,333,22,333,3333,33,122,13,4444,11,12
11	181	0.9807	125	181	333,11,111,222,333,22,333,3333,13,122,13,4444,11,22
			126	181	333,11,111,222,333,22,333,3333,23,122,11,4444,11,22
12	180	0.9803	128	180	333,11,111,222,333,22,333,3333,33,122,11,4444,11,22
13	179	0.9795	126	179	333,11,111,222,333,22,333,3333,33,122,13,4444,11,22
14	178	0.9784	125	178	333,11,111,222,333,22,333,3333,33,222,13,4444,11,22
15	177	0.9776	126	177	333,11,111,222,333,22,333,133,33,122,13,4444,11,22
16	176	0.9765	125	176	333,11,111,222,333,22,333,133,33,222,13,4444,11,22
17	175	0.9757	125	175	333,11,111,222,333,22,13,3333,33,122,11,4444,11,22
18	174	0.9749	123	174	333,11,111,222,333,22,13,3333,33,122,13,4444,11,22
19	173	0.9738	122	173	333,11,111,222,333,22,13,3333,33,222,13,4444,11,22
20	172	0.9730	123	172	333,11,111,222,333,22,13,133,33,122,13,4444,11,22
21	171	0.9719	122	171	333,11,111,222,333,22,13,133,33,222,13,4444,11,22
22	170	0.9708	120	170	333,11,111,222,333,22,13,133,33,222,33,4444,11,22
23	169	0.9693	121	169	333,11,111,222,333,22,33,133,33,222,13,4444,11,22
24	168	0.9681	119	168	333,11,111,222,333,22,33,133,33,222,33,4444,11,22
25	167	0.9663	118	167	333,11,111,222,33,22,13,133,33,222,33,4444,11,22
26	166	0.9650	116	166	333,11,11,222,333,22,13,133,33,222,33,4444,11,22
27	165	0.9637	117	165	333,11,111,222,33,22,33,133,33,222,33,4444,11,22
28	164	0.9624	115	164	333,11,11,222,333,22,33,133,33,222,33,4444,11,22
29	163	0.9606	114	163	333,11,11,222,33,22,13,133,33,222,33,4444,11,22
30	162	0.9592	115	162	333,11,11,222,33,22,33,133,33,222,13,4444,11,22
31	161	0.9580	113	161	333,11,11,222,33,22,33,133,33,222,33,4444,11,22
32	160	0.9557	112	160	333,11,11,222,33,22,33,333,33,222,13,4444,11,22
33	159	0.9546	110	159	333,11,11,222,33,22,33,333,33,222,33,4444,11,22

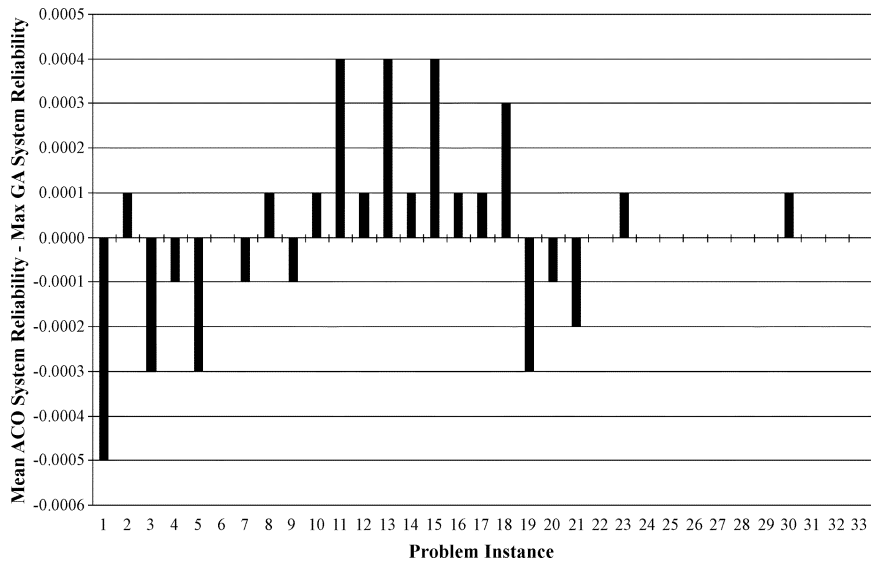


Fig. 2. Comparison of mean ACO with best GA performance over 10 seeds.

finds the best solution each and every run. While these differences in system reliability are not extreme, and there are many other factors to consider in choosing the best system design, it is advantageous to employ a search method that performs well

over different problem sizes and parameters. Moreover, any improvement that can be attained in system reliability while adhering to the design constraints is of some benefit, even if the reliability improvement realized is relatively small.

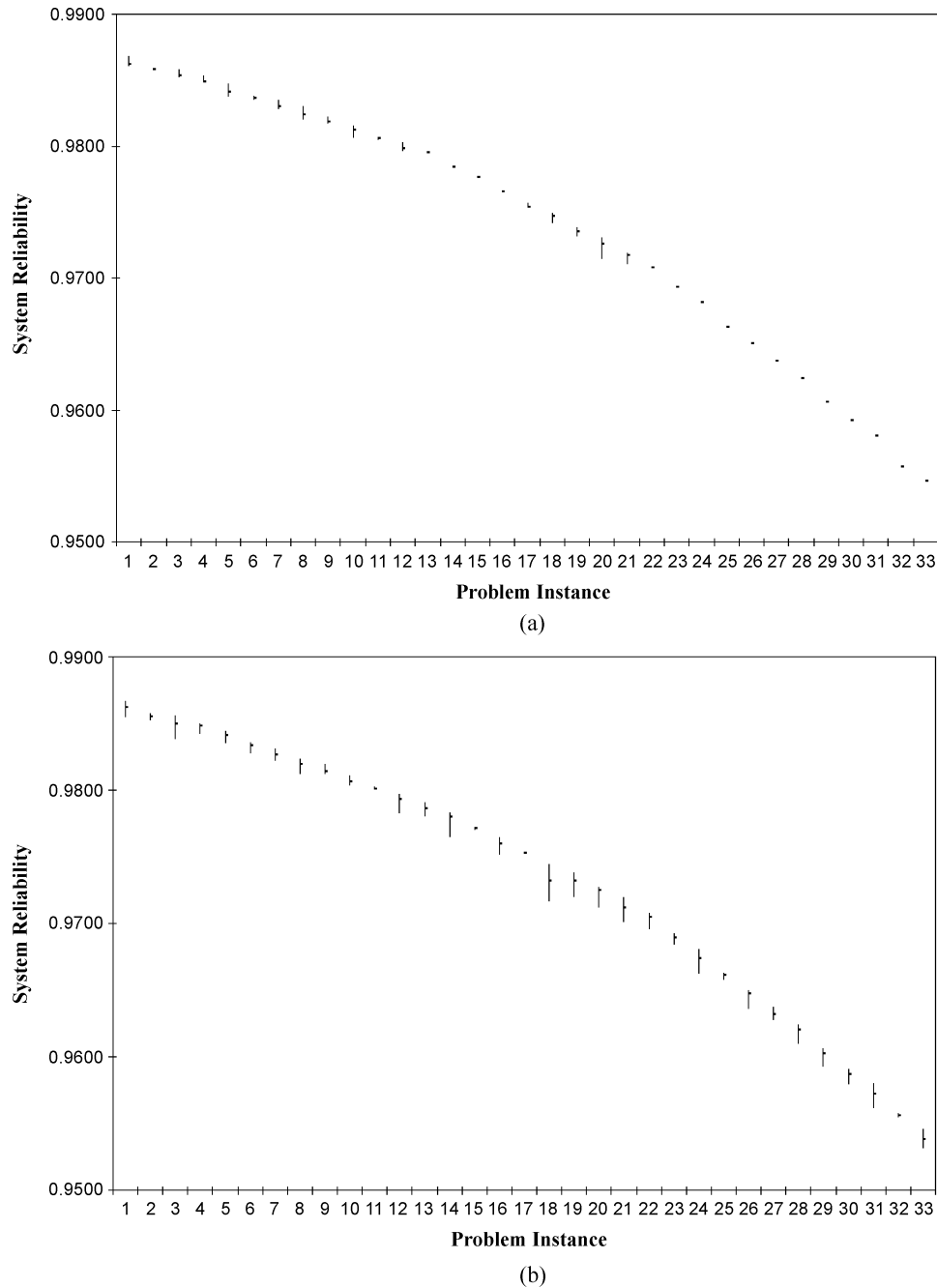


Fig. 3. Range of performance over 10 seeds with mean shown as horizontal dash. (a) ACO; (b) GA.

The configuration of the best solution; and its system reliability, cost, and weight for each of the 33 instances, are shown in Table II. For instances 6 & 11, two alternatives with different system cost, but with the same reliability and weight, are found. All but instance 33 involve mixing components within a subsystem, which is an indication that better designs can be identified by not restricting the search space to a single component type per subsystem.

It is difficult to make a precise computational comparison. CPU seconds vary according to hardware, software, and coding. Both the ACO and the GA generate multiple solutions during each iteration; therefore, the computational effort changes in direct proportion to number of solutions generated. The number of solutions generated in [7] (a population size of

40 with 1200 iterations) is about half of the ACO algorithm (a colony size of 100 with up to 1000 iterations). However, given the improved performance per seed of the ACO, a direct comparison per run is not meaningful. If the average solution of the ACO is compared to the best performance of GA; in 13 instances ACO is better, in 9 instances GA is better, and in the remaining instances (11) they are equal, as shown in Fig. 2. Because this is a comparison of average performance (ACO) versus best-of-ten performance (GA), the additional computational effort of the ACO is more than compensated for. In summary, an average run of ACO is apt to be as good, or better than the best of 10 runs of GA. The difference in variability over all 33 test problems between ACO and the GA is clearly shown in Fig. 3.

Given the well-structured neighborhood of the RAP, a meta-heuristic which exploits it is likely to be more effective, and more efficient, than one that does not. While the GA certainly performs well relative to previous approaches, the largely random mechanisms of crossover and mutation result in greater run to run variability than the ACO. Because the ACO shares the GA's attributes of flexibility, robustness, and implementation ease, and improves on its random behavior; it seems a very promising general method for other NP-hard reliability design problems such as those found in networks and complex structures.

ACKNOWLEDGMENT

The authors wish to thank the two reviewers and the associate editor for their careful reading and valuable suggestions to improve this paper.

REFERENCES

- [1] A. Bauer, B. Bullnheimer, R. F. Hartl, and C. Strauss, "Minimizing total tardiness on a single machine using ant colony optimization," *Central Eur. J. Oper. Res.*, vol. 8, no. 2, pp. 125–141, 2000.
- [2] R. Bellman and S. Dreyfus, "Dynamic programming and the reliability of multicomponent devices," *Oper. Res.*, vol. 6, pp. 200–206, 1958.
- [3] R. L. Bulfin and C. Y. Liu, "Optimal allocation of redundant components for large systems," *IEEE Trans. Rel.*, vol. R-34, no. 3, pp. 241–247, 1985.
- [4] B. Bullnheimer, R. F. Hartl, and C. Strauss, "Applying the ant system to the vehicle routing problem," in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, S. Martello, I. H. Osman, and C. Roucairol, Eds: Kluwer, 1999, pp. 285–296.
- [5] —, "An improved ant system algorithm for the vehicle routing problem," *Ann. Oper. Res.*, vol. 89, pp. 319–328, 1999.
- [6] M. S. Chern, "On the computational complexity of reliability redundancy allocation in a series system," *Oper. Res. Lett.*, vol. 11, pp. 309–315, 1992.
- [7] D. W. Coit and A. E. Smith, "Reliability optimization of series-parallel systems using a genetic algorithm," *IEEE Trans. Rel.*, vol. 45, no. 2, pp. 254–260, 1996.
- [8] —, "Design optimization to maximize a lower percentile of the system-time-to-failure distribution," *IEEE Trans. Rel.*, vol. 47, no. 1, pp. 79–87, 1998.
- [9] A. Colomi, M. Dorigo, V. Maniezzo, and M. Trubian, "Ant system for job-shop scheduling," *Belgian J. Oper. Res., Statist. Comp. Sci. (JORBEL)*, vol. 34, no. 1, pp. 39–53, 1994.
- [10] D. Costa and A. Hertz, "Ants can color graphs," *J. Oper. Res. Soc.*, vol. 48, pp. 295–305, 1997.
- [11] M. den Besteb, T. Stützle, and M. Dorigo, "Ant colony optimization for the total weighted tardiness problem," in *Proc. 6th Int. Conf. Parallel Problem Solving From Nature (PPSN VI)*, Berlin, 2000, pp. 611–620.
- [12] G. Di Caro and M. Dorigo, "Ant colonies for adaptive routing in packet-switched communication networks," in *Proc. 5th Int. Conf. Parallel Problem Solving From Nature (PPSN V)*, Amsterdam, The Netherlands, September 1998, pp. 673–682.
- [13] M. Dorigo, "Optimization, Learning and Natural Algorithms," Ph.D. Thesis, Politecnico di Milano, Italy, 1992.
- [14] M. Dorigo and G. Di Caro, "The ant colony optimization meta-heuristic," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds: McGraw-Hill, 1999, pp. 11–32.
- [15] M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999.
- [16] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *BioSystems*, vol. 43, pp. 73–81, 1997.
- [17] —, "Ant colony system: a cooperative learning approach to the travelling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.
- [18] M. Dorigo, V. Maniezzo, and A. Colomi, "Ant system: optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, and Cybernetics—Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.
- [19] D. E. Fyffe, W. W. Hines, and N. K. Lee, "System reliability allocation and a computational algorithm," *IEEE Trans. Rel.*, vol. R-17, no. 2, pp. 64–69, 1968.
- [20] L. M. Gambardella, E. Taillard, and G. Agazzi, "MACS-VRPTW a multiple ant colony system for vehicle routing problems with time windows," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds: McGraw-Hill, 1999, pp. 63–76.
- [21] 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 and Industrial Engineering*, vol. 24, no. 4, pp. 539–549, 1993.
- [22] P. M. Ghare and R. E. Taylor, "Optimal redundancy for reliability in series systems," *Oper. Res.*, vol. 17, pp. 838–847, 1969.
- [23] W. Kuo and V. R. Prasad, "An annotated overview of system-reliability optimization," *IEEE Trans. Rel.*, vol. 49, no. 2, pp. 176–187, 2000.
- [24] G. Levitin, A. Lisnianski, H. Ben-Haim, and D. Elmakis, "Redundancy optimization for series-parallel multi-state systems," *IEEE Trans. Rel.*, vol. 47, no. 2, pp. 165–172, 1998.
- [25] V. Maniezzo and A. Colomi, "The ant system applied to the quadratic assignment problem," *IEEE Trans. Knowledge and Data Engineering*, vol. 11, no. 5, pp. 769–778, 1999.
- [26] K. B. Misra and U. Sharma, "An efficient algorithm to solve integer-programming problems arising in system-reliability design," *IEEE Trans. Rel.*, vol. 40, no. 1, pp. 81–91, 1991.
- [27] Y. Nakagawa and S. Miyazaki, "Surrogate constraints algorithm for reliability optimization problems with two constraints," *IEEE Trans. Rel.*, vol. R-30, no. 2, pp. 175–180, 1981.
- [28] L. Painton and J. Campbell, "Genetic algorithms in optimization of system reliability," *IEEE Trans. Rel.*, vol. 44, no. 2, pp. 172–178, 1995.
- [29] L. Schoofs and B. Naudts, "Ant colonies are good at solving constraint satisfaction problems," in *Proc. 2000 Congress on Evolutionary Computation*, San Diego, CA, July 2000, pp. 1190–1195.
- [30] T. Stuetzle and M. Dorigo, "ACO algorithms for the quadratic assignment problem," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds: McGraw-Hill, 1999.
- [31] F. A. Tillman, C. L. Hwang, and W. Kuo, "Optimization techniques for system reliability with redundancy—a review," *IEEE Trans. Rel.*, vol. R-26, no. 3, pp. 148–155, 1977.
- [32] —, "Determining component reliability and redundancy for optimum system reliability," *IEEE Trans. Rel.*, vol. R-26, no. 3, pp. 162–165, 1977.
- [33] I. A. Wagner and A. M. Bruckstein, "Hamiltonian(t)—an ant inspired heuristic for recognizing Hamiltonian graphs," in *Proc. 1999 Congress on Evolutionary Computation*, Washington, D.C., July 1999, pp. 1465–1469.

Yun-Chia Liang received his first M.S. (Mechanical Engineering, 1996) from Carnegie Mellon University, a second M.S. (Industrial Engineering, 1999) from University of Pittsburgh, and the Ph.D. in Industrial and Systems Engineering from Auburn University (2001). His research interests are in application of meta-heuristics to optimization problems. Dr. Liang is currently an Assistant Professor in the Department of Industrial Engineering and Management at Yuan Ze University, Taiwan, R.O.C. He is a member of IEEE and IIE.

Alice E. Smith is Philpott-WestPoint Stevens Professor and Chair of the Industrial and Systems Engineering Department at Auburn University. Her research in analysis, modeling, and optimization of manufacturing processes and engineering design has been funded by the National Institute of Standards (NIST), NASA, Lockheed Martin, Adtranz (now Bombardier Transportation), the Ben Franklin Technology Center of Western Pennsylvania, and the National Science Foundation (NSF), from which she was awarded a CAREER grant in 1995 and an ADVANCE Leadership grant in 2001. International research collaborations have been sponsored by the federal governments of Japan, Turkey, United Kingdom and the U.S. Dr. Smith holds one U.S. patent and has authored over 100 publications in books, refereed proceedings and journals including articles in *IIE Transactions*, *IEEE TRANSACTIONS ON RELIABILITY*, *INFORMS Journal on Computing*, *International Journal of Production Research*, *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*, *Journal of Manufacturing Systems*, *The Engineering Economist*, and *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*. Dr. Smith holds editorial positions on *INFORMS Journal on Computing*, *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION* and *IIE Transactions*. Dr. Smith is a fellow of IIE; a senior member of 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 serves on the Educational Foundation Board of the Institute of Industrial Engineers.