

The Ant Colony Paradigm for Reliable Systems Design

Yun-Chia Liang

Department of Industrial Engineering and Management

Yuan Ze University

135 Yuan Tung Road

Chungli, Taoyuan 320, Taiwan, R.O.C.

Alice E. Smith

Department of Industrial and Systems Engineering

Auburn University

207 Dunstan Hall

Auburn, AL 36849 USA

I. Introduction

This chapter introduces a relatively new meta-heuristic for combinatorial optimization, the ant colony. The ant colony algorithm is a multiple solution global optimizer that iterates to find optimal or near optimal solutions. Like its siblings genetic algorithms and simulated annealing, it is inspired by observation of natural systems, in this case, the behavior of ants in foraging for food. Since there are many difficult combinatorial problems in the design of reliable systems, applying new meta-heuristics to this field makes sense. The ant colony approach with its flexibility and exploitation of solution structure is a promising alternative to exact methods, rules of thumb and other meta-heuristics.

The most studied design configuration of the reliability systems is a series system of s independent k -out-of- n :G subsystems as illustrated in Figure 1. A subsystem i is functioning properly if at least k_i of its n_i components are operational and a series-parallel system is where $k_i = 1$ for all subsystems. In this problem, multiple component choices are used in parallel in each subsystem. Thus, the problem is to select the optimal combination of components and redundancy levels to meet system level constraints while maximizing system reliability. Such a redundancy allocation problem (RAP) is NP-hard [6] and has been well studied (see Tillman, et al. [41] and Kuo & Prasad [25]).

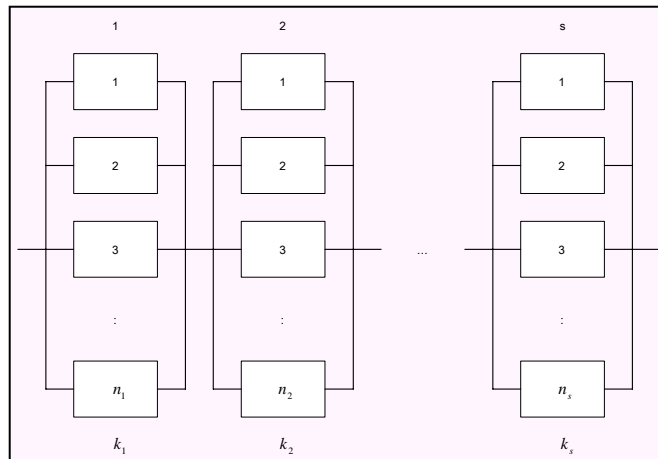


Fig. 1. Typical series-parallel system configuration.

Exact optimization approaches to the RAP include dynamic programming [2, 20, 34], integer programming [3, 22, 23, 32], and mixed-integer and nonlinear programming [42]. Because of the exponential increase in search space with problem size, heuristics have become a common alternative to exact methods. Meta-heuristics, in particular, are global optimizers that offer flexibility while not being confined to specific problem types or instances. Genetic algorithms (GA) have been applied by Painton & Campbell [36], Levitin et al. [26], and Coit & Smith [7, 8]. Kulturel-Konak et al. [24] use a Tabu search (TS) algorithm embedded with an adaptive version of the penalty function in [7] to solve RAPs. Three types of benchmark problems which consider the objectives of system cost minimization and system reliability maximization respectively were used to evaluate the algorithm performance. Liang and Wu [27] employ a variable neighborhood descent (VND) algorithm for the RAP. Four neighborhood search methods are defined to explore both the feasible and infeasible solution space.

Ant Colony Optimization (ACO) is one of the adaptive meta-heuristic optimization methods inspired by nature which include simulated annealing (SA), particle swarm optimization (PSO), GA and TS. ACO is distinct from other meta-heuristic methods in that it *constructs* a new solution set (colony) in each generation (iteration), while others focus on *improving* the set of solutions or a single solution from previous iterations. ACO was inspired by the behavior of physical ants. Ethologists have studied how blind animals, such as ants, could establish shortest paths from their nest to food sources and found that the medium used to communicate information among individual ants regarding paths is a chemical substance called pheromone. A moving ant lays some pheromone on the ground, thus marking the path. The pheromone, while dissipating over time, is reinforced if other ants use the same trail. Therefore, superior trails increase their pheromone level over time while inferior ones reduce to nil. Inspired by the behavior of ants, Marco Dorigo introduced the ant colony optimization approach in his Ph.D. thesis in 1992 [13] and expanded it in his further work including [14, 15, 18, 19]. The primary characteristics of ant colony optimization are:

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

ACO methods have been successfully applied to common combinatorial optimization problems including traveling salesman [16, 17], quadratic as-

signment [31, 40], vehicle routing [4, 5, 21], telecommunication networks [12], graph coloring [10], constraint satisfaction [38], Hamiltonian graphs [43] and scheduling [1, 9, 11]. A comprehensive survey of ACO algorithms and applications can be found in [19].

The application of ACO algorithms to reliability system problems was first proposed by Liang and Smith [28, 29], and then enhanced by the same authors in [30]. Liang and Smith employ ACO variations to solve a system reliability maximization RAP. Section III uses the ACO algorithm in [30] as a paradigm to demonstrate the application of ACO to RAP.

Thus far, the applications of ACO to reliability system are still very limited. Shelokar et al. [39] propose ant algorithms to solve three types of system reliability models: complex (neither series nor parallel), N-stage mixed series-parallel, and a complex bridge network system. In order to solve problems with different number of objectives and different types of decision variables, the authors develop three ant algorithms for single objective combinatorial problem, single objective continuous problem, and bi-objective continuous problem, respectively. The ant algorithm of single objective combinatorial version use the pheromone information only to construct the solutions, and no online pheromone updating rule is applied. Two local search methods, swap and random exchange, are performed to the best ant. For continuous problems, the authors divided the colony into two groups – global ants and local ants. The global ant concept can be considered as a pure GA mechanism since these ants apply crossover and mutation and no pheromone is deposited. Local ants are improved by a stochastic hill-climbing technique, and an improving ant can deposit the improvement magnitude of the objective on the trails. Lastly, a clustering technique and Pareto concept are combined with the continuous version of ant algorithms to solve bi-objective problems. The authors compared their algorithms with methods in the literature such as SA, a generalized Lagrange function approach, and a random search method. The results on four sets of test problems show the superiority of ACO algorithms.

Ouiddir et al. [35] develop an ACO algorithm for multi-state electrical power system problems. In this system redesign problem, the objective is to minimize the investment over the study period while satisfying availability or performance criteria. The proposed ant algorithm is based on the Ant Colony System (ACS) of [17] and [30]. A universal moment generating function is used to calculate the availability of the repairable multi-state system. The algorithm is tested on a small problem with five subsystems, each with four to six component options. Samrout et al. [37] apply ACO to determine the component replacement conditions in series-parallel systems minimizing the preventive maintenance cost. Three algorithms are proposed – two based on Ant System (AS) [18] and one based on ACS

[17]. Different transition rules and pheromone updating rules are employed in each algorithm. Local search is not used. Given different mission times and availability constraints, the performance of the ACO algorithms is compared with a GA from the literature. One of the AS algorithms and the ACS algorithm outperform the GA while the other AS algorithm is consistently outperformed by the GA. Nahas and Nourelfath [33] use an AS algorithm to optimize the reliability of a series system with multiple choices and budget constraints. Online pheromone updating and local search are not used. The authors apply a penalty function to determine the magnitude of pheromone deposition. Four examples with up to 25 component options are tested to verify the performance of the proposed algorithm. The computational results show that the AS algorithm is effective with respect to solution quality and computational expense.

The remaining chapter is organized as follows. Section II offers the notation list and defines the system reliability maximization RAP. A detailed introduction of an ant colony paradigm on solving RAP is provided in Section III using the work of Liang and Smith as a basis. Computational results on a set of benchmark problems are discussed in Section IV. Finally, concluding remarks are summarized in Section V.

II. Problem Definition

A. Notation

Redundancy Allocation Problem (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 -out-of- n : G	a system that functions when at least k of its n components function
R	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
\mathbf{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 that can be in parallel (user assigned)
k_i	minimum number of components in parallel required for subsystem i to function
$R_i(\mathbf{y}_i k_i)$	reliability of subsystem i , given k_i
$C_i(\mathbf{y}_i)$	total cost of subsystem i
$W_i(\mathbf{y}_i)$	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

Ant Colony Optimization (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}	$= \frac{1}{a_i}$, 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

B. Redundancy Allocation Problem

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 that can be relaxed using heuristics.

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

Subject to the constraints

$$\sum_{i=1}^s C_i(\mathbf{y}_i) \leq C, \quad (2)$$

$$\sum_{i=1}^s W_i(\mathbf{y}_i) \leq W, \quad (3)$$

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

$$k_i \leq \sum_{j=1}^{a_i} y_{ij} \leq n_{\max} \quad \forall i = 1, 2, \dots, s \quad (4)$$

Typical assumptions are:

- The states of components and the system are either operating 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 is assumed).
- Component attributes (reliability, weight and cost) are known and deterministic.

- The supply of any component is unconstrained.

III. Ant Colony Optimization Approach

This section is taken from the authors' earlier work in using the ant colony approach for reliable systems optimization [28, 29, 30]. The generic components of ant colony are each defined and the overall flow of the method is defined. These should be applicable, with minor changes, to many problems in reliable systems combinatorial design.

A. Solution Encoding

As with other meta-heuristics, it is important to devise a solution encoding that provides (ideally) a one to one relationship with the solutions to be considered during search. For combinatorial problems this generally takes the form of a binary or k-ary string although occasionally other representations such as real numbers can be used. For the RAP, each ant represents a design of an 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 are chosen from a_i available types of components. The a_i types are sorted 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 (that is, was left blank) 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 system design (that is, an ant) is an integer vector of length $n_{\max} \times s$.

B. Solution Construction

Also, as with other meta-heuristics, an initial solution set must be generated. For global optimizers the solution quality in this set is not usually important and that is true for the ant approach as well. In the ACO-RAP algorithm, ants use problem-specific heuristic information, denoted by η_{ij} , along with 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 calculated by equations 5 and 6, below. This

range of components encourages the construction of a solution that is likely to be feasible, that is, be reliable enough (satisfying the $k_i + 1$ lower bound) but not violate the weight and cost constraints (satisfying 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 III D).

The ACO problem specific heuristic chosen is $\eta_{ij} = \frac{r_{ij}}{c_{ij} + w_{ij}}$ where r_{ij} ,

c_{ij} , and w_{ij} represent the associated reliability, cost and weight of component j for subsystem i . This favors components with higher reliability and smaller cost and weight. Adhering to the ACO meta-heuristic concept, this is a simple and obvious rule. Uniform pheromone trail intensities for the initial iteration (colony of ants) are set over the component choices, that is,

$\tau_{i0} = \frac{1}{a_i}$. The pheromone trail intensities are subsequently changed as described in Section III E.

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

$$j = \begin{cases} \arg \max_{l \in AC} [(\tau_{il})^\alpha (\eta_{il})^\beta] & q \leq q_0 \\ J & q > q_0 \end{cases} \quad (5)$$

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

$$P_{ij} = \begin{cases} \frac{(\tau_{ij})^\alpha (\eta_{ij})^\beta}{\sum_{l=1}^{a_i} (\tau_{il})^\alpha (\eta_{il})^\beta} & j \in AC \\ 0 & \textit{Otherwise} \end{cases} \quad (6)$$

where α and β 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 uniform random number, and q_0 determines the relative importance of the exploitation of superior solutions versus the di-

versification 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, has the highest product of pheromone intensity and ratio of reliability to cost and weight. When $q > q_0$, the search favors more exploration as all components are considered for selection with some probability.

C. Objective Function

Fitness (the common term for the analogy to objective function value for nature inspired heuristics) plays an important role in the ant colony approach as it determines the construction probabilities for the subsequent generation. After solution u is constructed, the unpenalized reliability R_u is calculated using equation (1). For solutions with cost that exceeds C and / or weight that exceeds W , the penalized reliability R_{up} is calculated:

$$R_{up} = R_u \cdot \left(\frac{W}{W_u} \right)^\gamma \cdot \left(\frac{C}{C_u} \right)^\gamma \quad (7)$$

where the exponent γ is an amplification parameter and W_u and C_u are the 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 allows, search further into the infeasible region.

D. Improving Constructed Solutions Through Local Search

After an ant colony is generated, each ant is improved using local search. Local search is an optional, but usually beneficial, aspect of the ant colony approach that allows a systematic enhancement of the constructed ants. For the RAP, starting with the first subsystem, a chosen component type is deleted and a different 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 achieved, the new solution replaces the old one and the process continues until all subsystems have been searched. This local search does not require recalculating the system reli-

ability each time, only the reliability of the subsystem under consideration needs to be recalculated.

E. Pheromone Trail Intensity Update

The pheromone trail is a unique concept to the ant approach. Naturally, this idea is taken directly from studying physical ants and their deposits of the pheromone chemical. For the RAP, the pheromone trail update consists of two phases – online (ant-by-ant) updating and offline (colony) updating. Online updating is done after each solution is constructed and its purpose is to lessen the pheromone intensity of the components of the solution just constructed to encourage exploration of other component choices in the later solutions to be constructed. Online updating is by

$$\tau_{ij}^{new} = \rho \cdot \tau_{ij}^{old} + (1 - \rho) \cdot \tau_{i_0} \quad (8)$$

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

$$\tau_{ij}^{new} = \rho \cdot \tau_{ij}^{old} + (1 - \rho) \cdot \sum_{m=1}^E (E - m + 1) \cdot R_{mp} \quad (9)$$

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

F. Overall Ant Colony Algorithm

Generally, ant colony algorithms are similar to other meta-heuristics in that they iterate over generations (termed colonies for ACO) until some termination criteria are met. If an algorithm is elitist (as most genetic algorithms and ant colonies are) the best solution found is also contained in the final iteration (colony). The termination criteria are usually a combination of total solutions considered (or total computational time) and lack of best solution improvement over some number iterations. These are experimentally determined. Of course, there is no downside to running the ACO overly long except waste of computer time.

The flow of the ACO-RAP is as follows:

Set all parameter values and initialize the pheromone trail intensities
Loop
 Sub-Loop
 Construct an ant using the pheromone trail intensity and
 the problem-specific heuristic (eq.s 5, 6)
 Apply the online pheromone intensity update rule (eq. 8)
 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 (eq.s 1, 7), rank them and record the
 best feasible one
 Apply the offline pheromone intensity update rule (eq. 9)
Continue until a stopping criterion is reached

IV. Computational Experience

To show the effectiveness of the ant colony approach for reliable systems design results from [30] are given here. 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 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 colony are infeasible. This bi-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} and τ_{ij} are normalized between (0,1) before solution construction. 100 ants are used in each colony. The stopping criterion is when the number of colonies reaches 1000 or the best feasible solution has not changed for 500 consecutive colonies. This results in a maximum of 100,000 ants.

The 33 variations of the Fyffe et al. problem [20] as devised by Nakagawa and Miyazaki [34] were used to test the performance of ACO. In

this problem set $C = 130$ and W is decreased incrementally from 191 to 159. In [20] and [34], the optimization approaches required that identical components be placed in redundancy, however for the ACO approach, as in Coit and Smith [7], different component types are allowed to reside in parallel (assuming that a value of $n_{\max} = 8$ for all subsystems). This makes the search space size larger than 7.6×10^{33} . Since the heuristic benchmark for the RAP with component mixing is the GA of [7], it is chosen for comparison. Ten runs of each algorithm (GA and ACO) were made using different random number seeds for each problem instance.

The results are summarized in Table 1 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 shaded box shows the maximum reliability solution to an instance while considering all GA and ACO results. The ACO solutions are equivalent to or superior to the GA over 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 three instances in terms of the Max R measure (best over ten 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, finds the best solution each and every run. While these differences in system reliability are not large, it is beneficial to use a search method that performs well over different problem sizes and parameters. Moreover, any system reliability improvement while adhering to the design constraints is of some value, even if the reliability improvement realized is relatively small.

The best design and its system reliability, cost and weight for each of the 33 instances are shown in Table 2. For instances 6 and 11, two designs with different system costs but with the same reliability and weight are found. All but instance 33 involve mixing of components within a subsystem which is an indication that superior designs can be identified by not restricting the search space to a single component type per subsystem.

Table 1. Comparison of the GA [7] and the ACO over 10 random number seeds each for the test problems from [34]. These results are from [30].

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

Table 2. Configuration, reliability, cost and weight of the best solution to each problem. These results are from [30].

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

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 considered. The number of solutions generated in [7] (a population size of

40 with 1200 iterations) is about half of the ACO (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 over ten seeds is compared to the best performance of GA over ten seeds, in 13 instances ACO is better, in 9 instances GA is better and in the remaining instances (11) they are equal, as shown in Figure 2. Since this is a comparison of average performance (ACO) versus best performance (GA), the additional computational effort of the ACO is more than compensated for. In summary, an average run of ACO is likely to be as good or better than the best of ten runs of GA. The difference in variability over all 33 test problems between ACO and the GA is clearly shown in Figure 3.

Given the well-structured neighborhood of the RAP, a meta-heuristic that 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. Since 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.

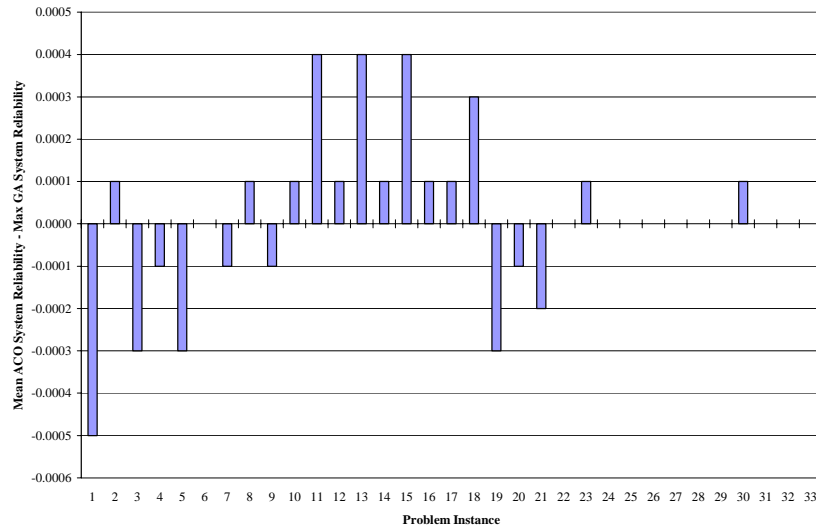
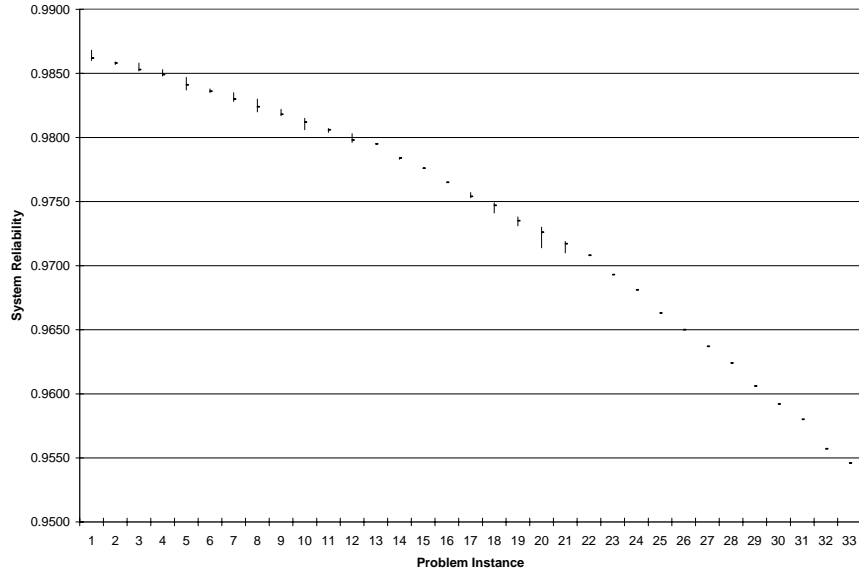
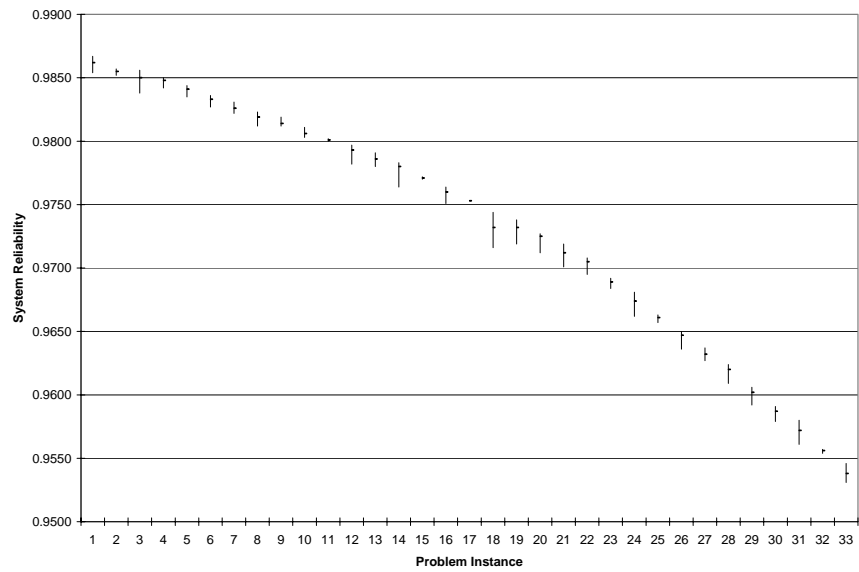


Fig. 2. Comparison of mean ACO with best GA performance over 10 seeds. These results are from [30].



(a) ACO



(b) GA

Fig. 3. Range of performance over 10 seeds with mean shown as horizontal dash. These results are from [30].

V. Conclusions

This chapter cites the latest developments of ACO algorithms to reliability system problems. The main part of the chapter gives details of a general ant colony meta-heuristic to solve the redundancy allocation problem (RAP) which was devised over the past several years by the authors and published in [28, 29, 30]. The RAP is a well known NP-hard problem that 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. Heuristic methods can overcome this limitation and offer a practical way to solve large instances of a relaxed RAP where different components can be placed in parallel. The ant colony algorithm for the RAP is shown to perform well with little variability over problem instance or random number seed. It is competitive with the best-known heuristics for redundancy allocation. Undoubtedly there will be much more work forthcoming in the literature that uses the ant colony paradigm to solve the many difficult combinatorial problems in the field of reliable system design.

References

1. Bauer A, Bullnheimer B, Hartl RF, Strauss C (2000) Minimizing total tardiness on a single machine using ant colony optimization. *Central European Journal of Operations Research* 8(2):125-141
2. Bellman R, Dreyfus S (1958) Dynamic programming and the reliability of multicomponent devices. *Operations Research* 6:200-206
3. Bulfin RL, Liu CY (1985) Optimal allocation of redundant components for large systems. *IEEE Transactions on Reliability* R-34(3):241-247
4. Bullnheimer B, Hartl RF, Strauss C (1999a) Applying the ant system to the vehicle routing problem. In: Voss S, Martello S, Osman IH, Roucairol C (eds) *Meta-heuristics: Advances and trends in local search paradigms for optimization*. Kluwer, pp 285-296
5. Bullnheimer B, Hartl RF, Strauss C (1999b) An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research* 89:319-328
6. Chern MS (1992) On the computational complexity of reliability re-

- dundancy allocation in a series system. *Operations Research Letters* 11:309-315
7. Coit DW, Smith AE (1996) Reliability optimization of series-parallel systems using a genetic algorithm. *IEEE Transactions on Reliability* 45(2):254-260
 8. Coit DW, Smith AE (1998) Design optimization to maximize a lower percentile of the system-time-to-failure distribution. *IEEE Transactions on Reliability* 47(1):79-87
 9. Colorni A, Dorigo M, Maniezzo V, Trubian M (1994) Ant system for job-shop scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science (JORBEL)* 34(1):39-53
 10. Costa D, Hertz A (1997) Ants can colour graphs. *Journal of the Operational Research Society* 48:295-305
 11. den Besten M, Stützle T, Dorigo M (2000) Ant colony optimization for the total weighted tardiness problem. *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN VI)*, LNCS 1917, Berlin, pp 611-620
 12. Di Caro G, Dorigo M (1998) Ant colonies for adaptive routing in packet-switched communication networks. *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature (PPSN V)*, Amsterdam, The Netherlands, pp 673-682
 13. Dorigo M (1992) Optimization, learning and natural algorithms. Ph.D. thesis, Politecnico di Milano, Italy
 14. Dorigo M, Di Caro G (1999) The ant colony optimization meta-heuristic. In: Corne D, Dorigo M, Glover F (eds) *New ideas in optimization*. McGraw-Hill, pp 11-32
 15. Dorigo M, Di Caro G, Gambardella LM (1999) Ant algorithms for discrete optimization. *Artificial Life* 5(2):137-172
 16. Dorigo M, Gambardella LM (1997) Ant colonies for the travelling salesman problem. *BioSystems* 43:73-81
 17. Dorigo M, Gambardella LM (1997) Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation* 1(1):53-66
 18. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics* 26(1):29-41

19. Dorigo M, Stützle T (2004) Ant colony optimization. The MIT Press, Cambridge
20. Fyffe DE, Hines WW, Lee NK (1968) System reliability allocation and a computational algorithm. *IEEE Transactions on Reliability* R-17(2):64-69
21. Gambardella LM, Taillard E, Agazzi G (1999) MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In: Corne D, Dorigo M, Glover F (eds) *New Ideas in Optimization*. McGraw-Hill, pp 63-76
22. Gen M, Ida K, Tsujimura Y, Kim CE (1993) Large-scale 0-1 fuzzy goal programming and its application to reliability optimization problem. *Computers and Industrial Engineering* 24(4):539-549
23. Ghare PM, Taylor RE (1969) Optimal redundancy for reliability in series systems. *Operations Research* 17:838-847
24. Kulturel-Konak S, Coit DW, Smith AE (2003) Efficiently solving the redundancy allocation problem using tabu search. *IIE Transactions* 35(6):515-526
25. Kuo W, Prasad VR (2000) An annotated overview of system-reliability optimization. *IEEE Transactions on Reliability* 49(2):176-187
26. Levitin G, Lisnianski A, Ben-Haim H, Elmakis D (1998) Redundancy optimization for series-parallel multi-state systems. *IEEE Transactions on Reliability* 47(2):165-172
27. Liang YC, Wu CC (2005) A variable neighborhood descent algorithm for the redundancy allocation problem. *Industrial Engineering and Management Systems* 4(1):109-116
28. Liang YC, Smith AE (1999) An ant system approach to redundancy allocation. *Proceedings of the 1999 Congress on Evolutionary Computation*, Washington, D.C., pp 1478-1484
29. Liang YC, Smith AE (2000) Ant colony optimization for constrained combinatorial problems. *Proceedings of the 5th Annual International Conference on Industrial Engineering – Theory, Applications and Practice*, Hsinchu, Taiwan, ID 296
30. Liang YC, Smith AE (2004) An ant colony optimization algorithm for the redundancy allocation problem (RAP). *IEEE Transactions on Reliability* 53(3):417-23

31. Maniezzo V, Colorni A (1999) The ant system applied to the quadratic assignment problem. *IEEE Transactions on Knowledge and Data Engineering* 11(5):769-778
32. Misra KB, Sharma U (1991) An efficient algorithm to solve integer-programming problems arising in system-reliability design. *IEEE Transactions on Reliability* 40(1):81-91
33. Nahas N, Nourelfath M (2005) Any system for reliability optimization of a series system with multiple-choice and budget constraints. *Reliability Engineering and System Safety* 87:1-12
34. Nakagawa Y, Miyazaki S (1981) Surrogate constraints algorithm for reliability optimization problems with two constraints. *IEEE Transactions on Reliability* R-30(2):175-180
35. Ouiddir R, Rahli M, Meziane R, Zeblah A (2004) Ant colony optimization or new redesign problem of multi-state electrical power systems. *Journal of Electrical Engineering* 55(3-4):57-63
36. Painton L, Campbell J (1995) Genetic algorithms in optimization of system reliability. *IEEE Transactions on Reliability* 44(2):172-178
37. Samrout M, Yalaoui F, Châtelet E, Chebbo N (2005) New methods to minimize the preventive maintenance cost of series-parallel systems using ant colony optimization. *Reliability Engineering and System Safety* 89:346-354
38. Schoofs L, Naudts B (2000) Ant colonies are good at solving constraint satisfaction problems. *Proceedings of the 2000 Congress on Evolutionary Computation, San Diego, CA*, pp 1190-1195
39. Shelokar P, Jayaraman VK, Kulkarni BD (2002) Ant algorithm for single and multiobjective reliability optimization problems. *Quality and Reliability Engineering International* 18:497-514
40. Stützle T, Dorigo M (1999) ACO algorithms for the quadratic assignment problem. In: Corne D, Dorigo M, Glover F (eds) *New ideas in optimization*. McGraw-Hill, pp 33-50
41. Tillman FA, Hwang CL, Kuo W (1977a) Optimization techniques for system reliability with redundancy - A review. *IEEE Transactions on Reliability* R-26(3):148-155
42. Tillman FA, Hwang CL, Kuo W (1977b) Determining component reliability and redundancy for optimum system reliability. *IEEE Transactions on Reliability* R-26(3):162-165

43. Wagner IA, Bruckstein AM (1999) Hamiltonian(t) - An ant inspired heuristic for recognizing Hamiltonian graphs. Proceedings of the 1999 Congress on Evolutionary Computation, Washington, D.C., pp 1465-1469