ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

A THESIS SUBMITTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

IN PARTIAL FULFILLMENT OF THE
REQUIREMENT FOR THE
MASTER DEGREE IN SYSTEM TECHNOLOGY
M.Eng.

BY
JIAN-HUA ZHAO

THE RELIABILITY OPTIMIZATION OF MECHANICAL SYSTEMS USING METAHEURISTIC APPROACH

MONTREAL, MARCH 16, 2005

CE MÉMOIRE A ÉTÉ ÉVALUÉ
PAR UN JURY COMPOSÉ DE :

Thien My Dao, directeur du Mémoire
Département de génie mécanique à l'École de technologie supérieure

Henri Champliaud, Président du jury
Département de génie mécanique à l'École de technologie supérieure

Zhaoheng Liu, codirecteur du Mémoire
Département de génie mécanique à l'École de technologie supérieure

Tony Wong, membre du jury
Département de génie de la production automatisée à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC
LE 16 FÉVRIER 2005
À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

OPTIMISATION DE FIABILITÉ DES SYSTÈMES MÉCANIQUES PAR L'APPROCHE MÉTAHEURISTIQUE

Jianhua Zhao

Sommaire

Le problème d'optimisation de fiabilité des systèmes mécaniques est un problème compliqué avec contraintes multicritères, dont la solution optimale est en générale un compromis. Le travail présenté dans ce mémoire se concentre sur l'optimisation de fiabilité des systèmes mécaniques en séries parallèles. Basée sur le ACSRAP (Ant Colony System for Redundancy Apportionment Problem), une nouvelle approche est présentée. Cette approche combine les caractéristiques de l'ACS avec des recherches locales. Donc il optimise la fiabilité globale du système tout en satisfaisant les contraintes en terme de coût, de poids et de volume. Les avantages sur la précision, l'efficacité, et la capacité de la nouvelle approche sont illustrés par les résultats de comparaison de la nouvelle technique avec ceux obtenues par d'autres approches. En outre, l'application de la technique sur une boite de transmission (Gear Train System) est aussi présenté pour montrer les procédures de l'application de la nouvelle technique sur les cas réels.

2005/3/28.

OPTIMISATION DE FIABILITÉ DES SYSTÈMES MÉCANIQUES PAR L'APPROCHE MÉTAHEURISTIQUE

Jianhua Zhao

SOMMAIRE

Une approche métaheuristique basée sur le système de colonie de fourmis (SCF) a été développée avec succès pour résoudre des problèmes d'optimisation multicritères de fiabilité des systèmes. La solution des problèmes visés par cette approche consiste à identifier, dans un système en série parallèle, le niveau de fiabilité de chacun des composants et le niveau de redondance afin d'optimiser la fiabilité globale du système tout en satisfaisant les contraintes de coût, de poids et de volume. Ce sont des problèmes très communs et réalistes que l'ingénieur rencontre souvent dans la conception des systèmes mécaniques, électroniques et industriels. Il devient de plus en plus important de développer les solutions efficaces à ces problèmes puisque beaucoup de systèmes réels deviennent plus complexes avec le niveau de fiabilité plus exigent et la période de développement allouée plus courte. L'algorithme multi objectif développé dans le cadre de ce projet offre des avantages distincts à ces problèmes et cette approche développée peut être généralement utilisée comme une méthode générique pouvant optimiser une large gamme de problèmes différents sans nécessiter de changements profonds dans l'algorithme employé. Le nouvel algorithme conçu utilise plusieurs techniques telles que la recherche probabiliste, la formulation des fonctions objectives multiples, les avancements locaux et la méthode de pénalité dynamique. L'application de cet algorithme dans la solution des problèmes de répartition de redondance (en anglais « Redundancy Apportionment Problems (RAP ») des systèmes en série parallèle, tels que la conception de réducteur de transmission, démontre que la configuration optimale du système est obtenue plus fréquemment et plus rapidement qu'avec d'autres approches heuristiques. Par conséquent, l'application de l'approche proposée aux problèmes d'optimisation de fiabilité des systèmes mécaniques annonce le grand potentiel et permet de développer un outil puissant et économique pour résoudre des problèmes d'optimisation complexes dont les concepteurs/chercheurs doivent constamment faire face.

Mots-clés : optimisation de fiabilité, système série-parallèle, répartition de redondance, système de colonie de fourmis, optimisation multiobjective, conception de réducteur de transmission.

Jianhua Zhao	Thien-My Dao Directeur de mémoire

RELIABILITY OPTIMIZATION OF MECHANICAL SYSTEM WITH METAHEURISTIC APPROCHES

Jianhua Zhao

ABSTRACT

The multiobjective Ant Colony System (ACS) meta-heuristic has been developed successfully to provide a solution for the reliability optimization problems of series-parallel system and has been demonstrated its application to the reliability design of gearbox. The problems involve the selection of components with multiple choice and redundancy levels that produce maximum benefit, and are subject to the cost and weight constraints at the system level. These are very common and realistic problems involving conception design of engineering system and reliability engineering. It is becoming increasingly important to develop efficient solutions to these problems because many mechanical and electrical systems are becoming more complex, even as development schedules get shorter and reliability requirements become very stringent. The multiobjective Ant Colony System algorithm offers distinct advantages to these problems compared to alternative optimization methods, and can be applied to a more diverse problem domain with respect to the type or size of the problems. Through the combination of probabilistic search, multi-objective formulation of local moves and the dynamic penalty method, the multiobjective ACSRAP, which performs very well on the redundancy apportionment problems (RAP) of the series-parallel k-out-of-n : G subsystem and reliability design of gear box, allows us to obtain an optimal design solution very frequently and more quickly than with other heuristic approaches. Therefore, the use of these techniques to the reliability optimization problems of mechanical systems announces great potential and makes it possible to develop a powerful and economic tool for which the designers always seek.

Keywords — Reliability optimization, Series-parallel system, Redundancy apportionment, Ant Colony System, multiobjective Optimization, gearbox design

Jianhua Zhao	Thien-My Dao Directeur de mémoire

RÉSUMÉ DU MÉMOIRE EN FRANÇAIS

La conception des systèmes mécaniques avec un haut niveau de fiabilité à un coût optimal est un des facteurs importants pour la performance globale de ces systèmes. Cette performance à son tour a un impact certain sur la productivité et le coût opérationnel des systèmes industriels auxquels les systèmes mécaniques font partie. Pour augmenter la fiabilité globale d'un système, on peut employer un composant plus fiable, ajouter les composants comme redondance en parallèle ou faire la combinaison des deux approches mentionnées. Le problème de répartition avec redondance, en anglais « Redundancy Apportionment Problems (RAP) », est un sujet très répandu dans la conception optimale et dans la fiabilité des systèmes mécaniques, électriques et industriels. La solution du problème consiste à identifier la combinaison optimale des composants et le niveau de leur redondance dans chacun des sous-systèmes afin d'optimiser la fiabilité globale du système tout en satisfaisant les contraintes de conception (coût, poids, volume, etc.). Le RAP pour systèmes mécaniques est un problème complexe en raison de la présence des objectifs contradictoires, tels que minimiser le coût, le poids ou le volume du système considéré tout en maximisant simultanément la fiabilité du système. Il devient de plus en plus important de développer les solutions efficaces à ces problèmes puisque beaucoup de systèmes réels deviennent plus complexes avec le niveau de fiabilité plus exigeant et la période de développement allouée plus courte.

C'est dans cet esprit que le projet de recherche, dans le cadre de ce mémoire, vise à développer une approche métaheuristique pour obtenir la solution optimale du RAP. Le travail de recherche se limite à des systèmes mécaniques dont le modèle physique est représenté par un système en série parallèle. Les méthodes métaheuristiques sont choisies pour résoudre le RAP parce qu'elles peuvent optimiser une large gamme de problèmes différents, sans nécessiter de changements profonds dans l'algorithme employé. En pratique, les métaheuristiques sont surtout utilisées pour des problèmes ne pouvant pas être optimisés par des méthodes mathématiques. Le but principal de cette recherche est de

développer un algorithme métaheuristique basé sur le système de colonie de fourmis (en anglais Ant Colony System - ACS) permettant d'identifier le niveau de fiabilité de chacun des composants (en parallèle) et le niveau de redondance afin d'optimiser la fiabilité globale du système tout en satisfaisant les contraintes en termes de coût, de poids et de volume.

Le texte du mémoire a commencé par une introduction sur l'idée directrice et les objectifs principaux de la recherche proposée et la justification de l'utilisation du système de colonie de fourmis (Ant Colony System - ACS) pour résoudre le problème visé.

Le mémoire est présenté en cinq chapitres. La formation du problème d'optimisation de fiabilité des systèmes en série parallèle est présentée au chapitre 1. Dans ce même chapitre, une révision est faite sur différentes techniques utilisées pour résoudre le problème RAP. Les méthodologies et les caractéristiques des approches traditionnelles, comprenant la programmation mathématique telle que la programmation dynamique (DP) et la programmation du nombre entier (PE), des approches heuristiques (utilisées dans la programmation non linéaire) et des approches de l'optimisation multicritère ont été présentées. En raison de la difficulté de calcul, qui augmente exponentiellement en fonction de la taille du problème et du nombre de contraintes, les approches traditionnelles limitent artificiellement l'espace de recherche à la solution où seulement un type de composant est choisi pour chaque sous-système, et alors, le même type est employé pour fournir la redondance. Cependant, ces restrictions sont nécessaires pour l'application des stratégies d'optimisation, mais pas pour le problème réel de conception de technologie. En pratique, différents composants, exécutant la même fonction, peuvent être employés dans un système pour augmenter la fiabilité. Il est possible d'obtenir de meilleures solutions en détendant la restriction. Après l'analyse des inconvénients des approches traditionnelles, quelques approches métaheuristiques ont été présentées en ce chapitre. Par exemple, l'algorithme génétique (GA) utilise la recherche probabiliste pour surmonter les inconvénients des approches traditionnelles et donne d'excellents résultats pour le RAP.

Cependant, GA emploie une stratégie d'amélioration exigeant l'évaluation des solutions éventuelles avec beaucoup de générations. Ceci donne l'effort significatif du calcul pour des problèmes de grande échelle. Par conséquent, une approche plus efficace est souhaitable pour trouver la solution du problème posé. À la fin de ce chapitre, le système de colonie de fourmis (Ant Colony System ACS) est présenté comme un nouvel algorithme métaheuristique et constructif pour résoudre les problèmes de RAP et la justification d'utilisation du système de fourmis (Ant Colony System ACS) est explicitée.

Au chapitre 2 du mémoire, une revue complète est donnée sur le mécanisme et les procédures de l'algorithme d'ACS (Ant Colony System). Différentes versions des algorithmes avec des systèmes de colonie de fourmis et des techniques utilisées dans les algorithmes d'ACS sont présentées en ce chapitre. La relation entre les algorithmes d'ACS et les techniques existantes est également présentée. L'approche d'ACS est basée sur le système de fourmis qui imite le comportement de vraies fourmis recherchant de la nourriture. Les fourmis échangent leurs informations sur les sources de la nourriture, en déposant une essence aromatique appelée phéromone, tout au long de leur trajet. Avec le temps, les courts chemins directs, menant du nid à une source de nourriture, sont plus fréquentés que les plus longs chemins. En conséquence, les chemins directs sont marqués par plus de phéromone, ce qui attire en revanche plus de fourmis et fait agrandir les chemins de phéromone. Le processus est ainsi caractérisé par des itérations de rétroaction positive, où la probabilité pour que les fourmis choisissent un chemin spécifique augmente avec le nombre de fourmis qui ont précédemment choisi ce même chemin. Dans l'algorithme d'ACS, les fourmis artificielles imitent non seulement le comportement décrit, mais appliquent également l'information heuristique additionnelle et le problème spécifique.

Dans le passé, l'approche d'ACS a été appliquée et a fourni des solutions pour différents problèmes combinatoires NP durs. Les chercheurs ont proposé des solutions au RAP avec l'approche d'ACS. Cependant, le niveau de redondance de chaque sous-système peut augmenter la difficulté de calcul. Dans les travaux antérieurs, l'approche d'ACS a été appliquée aux systèmes en série où seulement un composant avec de multiples choix est employé dans chaque sous-système et seulement une contrainte du système (par exemple, le critère de coût) est considérée. D'autres chercheurs ont utilisé la stratégie en réduisant artificiellement les niveaux de redondance de chaque sous-ensemble et en améliorant alors toutes les solutions avec des techniques locales de recherche pour alléger les difficultés de calcul. Ces algorithmes peuvent fournir d'excellents résultats pour les systèmes relativement simples, mais l'efficacité de l'algorithme devient une problématique pour le cas des systèmes plus complexes. Dans un tel cas, des mesures doivent être prises pour préserver la caractéristique constructive de l'approche d'ACS en évitant de trop compter sur l'amélioration de solutions par des recherches locales.

Pour surmonter les inconvénients de la méthodologie d'ACS mentionnés précédemment dans la recherche de la solution de RAP, il est impératif de développer un nouvel algorithme multicritère par l'approche d'ACS qui permet de résoudre le RAP pour un système en série parallèle.

Le chapitre 3 du mémoire est la partie centrale des travaux de recherche de ce projet. Une technique baptisée ACSRAP (Ant. Colony System for Redundancy Apportionment Problem) multicritère a été développée et présentée dans ce chapitre. C'est une technique hybride basée sur les caractéristiques d'ACS et combinée à des recherches locales. La procédure de l'algorithme d'ACSRAP multicritère consiste en six étapes suivantes :

i) Initialiser la traînée de phéromone et la probabilité de transition en utilisant l'information heuristique. La phéromone initiale est définie comme la probabilité de choisir un composant dans chaque sous-système sans considérer l'information heuristique. L'information heuristique locale est le choix du meilleur composant avec un nombre minimum de niveaux de redondance.

ii) Produire la colonie de fourmis et construire la solution par règle de transition d'état et la politique déterministe de mouvement local avec la formulation multicritère et le premier type de recherche locale. Après avoir choisi le minimum de composants dans chaque sous-système, la fiabilité, le coût et le poids de chaque sous-système sont calculés respectivement. Une politique déterministe de mouvement local est alors employée pour choisir le sous-système à améliorer et le composant dans le sous-système. Les techniques de recherche locale sont utilisées ensuite pour améliorer la configuration du sous-système. Deux types de recherche locale sont utilisés dans l'algorithme développé. Pour le premier type de recherche locale, lorsque la redondance dans un sous-système excède un niveau donné, nous choisissons le meilleur composant et remplaçons les autres composants dans le même sous-système jusqu'à ce que ce sous-système emploie un budget minimum de ressources pour obtenir une meilleure fiabilité. L'autre type de recherche locale est exécuté après que les solutions soient établies dans chaque itération. Le voisinage de la meilleure solution faisable est exploré par la stratégie de l'échange 2-positions pour chaque sous-système.

iii) Appliquer le deuxième type de recherche locale (optionnel).

iv) Évaluer toutes les solutions avec la fonction dynamique de pénalité et appliquer la règle de mise à jour à la traînée de phéromone et à la probabilité de transition.

v) Stocker un certain nombre de fourmis faisable et enregistrer la meilleure solution faisable pendant l'itération pour augmenter les traînées de phéromone de bons composants. Cette stratégie d'élitiste est très utile pour construire la bonne configuration du système.

vi) Répéter les étapes ii) à v) jusqu'à ce que les critères d'arrêt soit rencontrés.

Les comparaisons des procédures d'ACSRAP multicritères avec les autres approches existantes sont aussi présentées au chapitre 3.

Au chapitre 4 du mémoire, nous avons démontré l'application de la nouvelle technique développée pour obtenir les solutions optimales dans un problème de RAP représenté par un système en série parallèle. Dans ce chapitre, les effets des paramètres tels que le nombre de fourmis dans une colonie, l'information heuristique locale, le facteur d'amplification du phéromone, la mémoire à long terme, la persistance de traînée de phéromone, la fonction de pénalité et différentes stratégies de recherche locale sont présentés et analysés pour l'algorithme développé. Les expériences démontrent que le choix des paramètres appropriés dans l'algorithme d'ACSRAP, comme tous les autres algorithmes métaheuristiques, joue toujours un rôle important pour obtenir une solution de qualité satisfaisante.

Après le choix des paramètres appropriés, les comparaisons de performance du nouvel algorithme d'ACSRAP multicritère avec l'algorithme génétique pour RAP (GARAP de Coit & Smith) et l'algorithme de colonie de fourmis existant (ACO-RAP de Liang & Smith) sont présentées.

En ce qui concerne la comparaison de l'ACSRAP multicritère avec GARAP, les résultats statistiques ont démontré que l'ACSRAP a généralement donné des solutions avec une fiabilité plus élevée et une variation inférieure au GARAP. En ce qui a trait à la comparaison de l'ACSRAP multicritère avec l'ACORAP, on a trouvé que l'exécution globale de fiabilité d'ACSRAP est semblable à l'ACO-RAP de Liang et Smith, mais l'ACSRAP multicritère a réduit le nombre d'itérations approximativement de 37,5 % par rapport à GARAP et d'environ 233 % par rapport à l'ACO-RAP. Par conséquent, la comparaison des résultats démontre un avantage prometteur du nouvel algorithme développé en termes de précision de la solution obtenue ainsi qu'au niveau de son efficacité et de sa capacité de traiter des problèmes à grande échelle.

Le chapitre 5 du mémoire présente une application de l'algorithme d'ACSRAP multicritère à un cas réel en ingénierie : l'optimisation de la fiabilité d'un réducteur de transmission (Gear Train System). C'est une démonstration de la capacité pratique de l'ACSRAP multicritère. Dans cette application, nous montrons comment la nouvelle technique pourrait être appliquée pour les problèmes de l'optimisation multicritère. Le modèle d'un réducteur de transmission est formulé comme étant un système en série parallèle. La présentation des résultats démontre que la nouvelle technique proposée pourrait être utilisée comme un outil pratique d'optimisation de conception des systèmes mécaniques à un niveau de fiabilité spécifié. L'approche interactive d'ACSRAP multicritère développée dans ce chapitre a allégé les difficultés liées aux méthodes classiques qui ont besoin d'information, de préférence, et de prétentions restrictives. Alors c'est une approche flexible, systématique et surtout applicable aux problèmes d'optimisation multicritère (combinés). Les résultats obtenus avec la nouvelle technique développée ont démontré que la configuration optimale du système est obtenue plus fréquemment et plus rapidement qu'avec d'autres approches heuristiques. Par conséquent, l'application de l'approche proposée aux problèmes d'optimisation de fiabilité des systèmes mécaniques annonce son grand potentiel et permet de développer un outil puissant et économique pour résoudre des problèmes d'optimisation complexes auxquels les concepteurs et les chercheurs doivent constamment faire face.

En conclusion, les contributions principales de cette recherche peuvent être résumées comme suit :

- L'algorithme multicritère par le système de colonie de fourmis développé dans ce projet est robuste pour résoudre des problèmes combinatoires (NP-hard). On propose une nouvelle approche d'ACS pour traiter le RAP représenté par un système général en série parallèle de k-dehors--n de G : sous-système. La nouvelle approche est très générale dans le sens qu'il n'y a aucune limite au nombre de types de composants et aux niveaux de redondance.

- La formulation multicritère de la stratégie du mouvement local dans la colonie de fourmis est efficace pour construire la configuration de sous-ensembles.
- L'ACS multicritère développé permet d'identifier les solutions infaisables dans la recherche et traite ces solutions infaisables par une fonction adaptative de pénalité. Cette fonction de pénalité intègre dynamiquement la fonction objective globale avec le résultat de recherche locale et la limite de contraintes.
- La stratégie d'élitiste en employant la mémoire à long terme pour stocker davantage les fourmis faisables augmente la chance du bon choix des composants.
- L'approche interactive d'ACSRAP multicritère allège les difficultés liées aux méthodes classiques qui ont besoin d'information, de préférence, et de prétentions restrictives. C'est une approche flexible, systématique et surtout applicable aux problèmes d'optimisation multicritère (combinés).
- Un article de recherche intitulé « Reliability optimization using multiobjective ant colony system approaches » a été accepté par la revue internationale «Journal of Reliability Engineering and System Safety » en novembre 2004.

Pour la suite du présent travail de recherche, l'approche de la multiple colonie en se basant sur l'ACS avec le calcul parallèle pourrait être considérée comme une bonne direction pour résoudre les problèmes d'optimisation multicritère. De plus, les phéromones différentes et les colonies hétérogènes peuvent être utilisées selon le cas afin de s'adapter aux diverses caractéristiques des problèmes retenus. Il convient de noter que l'algorithme d'ACS proposé dans ce document est plutôt simple. Certaines techniques normalement utilisées dans les problèmes complexes, telles que la liste de candidats ou d'autres techniques de recherche locale, ne sont pas incorporées dans cette recherche. L'algorithme pourrait être amélioré en termes de l'efficacité en considérant l'addition de ces mesures à l'algorithme d'ACS présenté dans ce mémoire.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Thien-My Dao and my co-advisor, Dr. Zhaoheng Liu for their invaluable guidance and excellent assistance during the course of this research. Working with them is very rewarding in various aspects not only limited to this research.

I am deeply grateful to all the professors who have so generously provide information and assistance required during my course study and this research. Without their help, this work would not have been possible.

My thanks also go to the expert reviewers who took the time to examine and critique the work in manuscript form.

Very special thanks go to my parents. I have been greatly influenced by them and their attitudes have always played an important role in my life.

Finally, I would like to express my deepest thanks to my wife and my daughter. Without their endless love, support and encouragement, this thesis could have never been completed. I wish to delicate this piece of research to my wife, *Meng Wu* and our lovely daughter, *Tracy*, who spreads joyful moments over our daily life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURES

α	Relative influence of the pheromone trail
β	Relative importance of heuristic information
τ_{ij}	Pheromone trail for the combination (i, j)
τ_b	Initial quantity of pheromone trail for the "blank" combination
$\Delta\tau_{ij}^a$	Quantity of pheromone trail added to τ_{ij} by a^{th} ant
$\Delta\tau_{ij}^e$	Quantity of pheromone trail added to τ_{ij} by elitist ants
e	Number of elitist ants
$\Delta\tau_{ij}^f$	Quantity of pheromone trails laid by feasible ants in RAP
η_{ij}	Heuristic information for the combination (i, j)
ρ	Persistence of pheromone trail. $0 \leq \rho \leq 1$
q	*A* uniformly generated random number. $0 \leq q \leq 1$
q_0	Relative influence of exploitation versus exploration
Q	Amount of pheromone trail deposited by an ant
P_{ij}	Transition probability for combination of component type j in subsystem i
NA	Number of ants
NA_{feas}	The number of feasible ants in each iteration in RAP
NC	Number of iterations
$NC_{\max}$	Maximum number of iterations (user defined)
Rs	System reliability
Cs	System cost
Ws	System weight
$Cs_{\max}$	Constraint of system cost
$Ws_{\max}$	Constraint of system weight
$Rs_{\min}$	Constraint of system reliability

i	Index of subsystem
j	Index of component type
k	Index of redundancy level
s	Number of subsystems in the system
m_i	Number of available component types in subsystem i ($i = 1, 2, \ldots, s$)
N_i	A set of component types for choice $N_i = [1, 2 \ldots m_i]$
N	$= [N_i, N_2, \ldots, N_s] = [1, 2 \ldots m_1;\ 1, 2 \ldots m_2;\ \ldots 1, 2 \ldots m_s]$
x_{ki}	Component type assigned to position k of subsystem i, $x_{ki} \in (1,2,\ldots,m_i, m_{i+1})$
n_i	Total number of redundant components used in subsystem i
$\mathbf{n}$	$= (n_1, n_2, \ldots, n_i)$
p_i	Minimum components in parallel required for subsystem i to function
PN	Maximum number of components in parallel (user defined)
r_{ij}	Reliability of the j^{th} component type for subsystem i
c_{ij}	Cost of the j^{th} component type for subsystem i.
w_{ij}	Weight of j^{th} component type for subsystem i
v	An index represents the used budget of system resource
$R_i(\boldsymbol{x})$	Reliability of subsystem i
$C_i(\boldsymbol{x})$	Total cost of subsystem i
$W_i(\boldsymbol{x})$	Total weight of subsystem i
R_{as}	The unpenalized system reliability of a^{th} ant solution
R_{ap}	The penalized objective function value of a^{th} ant solution
R_{all}	The unpenalized solution value of the best solution found in each iteration
R_{best}	The value of the best feasible solution found in each iteration
C_{as}	The unpenalized system reliability of the a^{th} ant solution
C_{ap}	The penalized objective function value of a^{th} ant's solution
C_{all}	The unpenalized solution value of the best solution found
C_{best}	The value of the best feasible solution found in each iteration

W_{as}	The unpenalized system reliability of the a^{th} ant solution
W_{ap}	The penalized objective function value of the a^{th} ant's solution
W_{all}	The unpenalized solution value of the best solution found
W_{best}	The value of the best feasible solution found in each iteration
NFT	Near feasibility Threshold
NFT_c	Near feasibility threshold for cost constraint
NFT_w	Near feasibility threshold for weight constraint
NFT_R	Minimum Near feasibility threshold of reliability among the feasible solutions in each iteration
NFT_{cmax}	Maximum near feasibility threshold of cost among the feasible solutions in each iteration
NFT_{wmax}	Maximum near feasibility threshold of weight among the feasible solutions in each iteration
NFT_{Rmin}	Minimum near feasibility threshold of weight among the feasible solutions in each iteration
ΔW_s	Magnitude of the cost constraints violations for ant's solution
ΔC_s	Magnitude of the weight constraints violations for ant's solution
ΔR_s	Magnitude of the reliability constraint violations for ant's solution
γ_1	Amplification parameter for cost constraints violations in penalty function
γ_2	Amplification parameter for weight constraints violations in penalty function
h	The thickness of the gear
T_w	Numbers of teeth on the wheel
T_p	Numbers of teeth on the pinion
ω_w	Speed of the pinion (rpm)
ω_p	Speed of the wheel (rpm)
GP_{ij}	The j^{th} type of gear pair to be used in stage i
G_i^1	The first gear in gear pair GP_{ij}

G_i^2	The second gear in gear pair GP_{ij}
f_{Rs}	Achievement factor to the objective of maximizing system reliability
f_{Cs}	Achievement factor to the objective of minimizing system cost
f_{Ws}	Achievement factor to the objective of minimizing system weight
k-out-of-n G subsystem:	A subsystem with n components is good or functional if and only if *k* components in the subsystem are good or functional
ACS	Ant Colony System
DP	Dynamic programming
IP	Integer programming
GA	Genetic algorithm
RAP	Redundancy apportionment problem
SA	Simulated annealing
TS	Tabu search
TSP	Traveling salesman problem

INTRODUCTION

Reliability is of critical importance in various types of electrical and mechanical systems. A well-known and complex reliability optimization problem is the redundancy apportionment problem (RAP) for series-parallel systems which can be identified as the selection of the optimal combination of component type and redundancy level for each subsystem in order to meet various objectives given constraints on the overall system. The problem can become complicated due to the presence of multiple conflicting objectives, such as minimizing the system cost and system weight or volume, while simultaneously maximizing the system reliability.

Many different formulations and different optimization approaches have been used in solving RAP problem. Due to the computational difficulty, which increases exponentially in terms of the problem size and the number of constraints Traditional approaches usually restrict artificially the search space to solution where only one component type can be chosen for each subsystem, and then the same type can be used to provide redundancy. After such restrictions, mathematical programming techniques, such as dynamic programming, Integer Programming, mixed integer and nonlinear programming, and multiobjective approaches are used to provide solutions. But such restrictions are not necessary for practical engineering problem, where different components can be used within each subsystem to provide high reliability. For example, multi-speed gearboxes use different gear pairs in each stage, and power plants can uses Gas turbines and steam turbines in parallel as prime movers. In practice, it is possible to obtain better solutions by relaxing the restriction.

Genetic Algorithms (GA) use probabilistic search to overcome the shortcomings of the traditional approaches and has obtained excellent results for RAP. However, GA uses an improving strategy requiring the evaluation of prospective solutions over many generations. This results in significant computational effort for large-scale problems.

Therefore, a more efficient and effective approach is desirable.

In this research, we focus on the development of heuristic methods and meta-heuristic algorithms for the reliability and redundancy apportionment problems (RAP) of complex electromechanical systems. Ant Colony System (ACS) is proposed as optimization methodology for RAP, and thus gives another evidence for its versatility. ACS is based on the Ant System that imitates the behavior of real ants when search food. Ants communicate information about food sources via the quantity of an aromatic essence called pheromone, which the ants deposit as they move along. Over time, the short direct paths leading from the nest to a food source are more frequented than longer paths. As a result, the direct paths are marked with more pheromone, which in turn attracts more ants to follow the paths and make corresponding pheromone trails grow faster. The process is thus characterized by a positive feedback loop, where the probability with which ants choose a path increases with the number of ants that previously chose that same path.

Some researchers tried to solve the RAP with ACS approach. Since implementing ACS for the RAP requires successive "moves" on different redundancy levels of each subsystem with different component choices to construct solutions, and high redundancy level results in significant computational effort and difficulty, previous researchers either consider only series systems without redundancy, or artificially reduce the redundancy levels of each subsystem and then improve all the solutions costly with local search techniques. Such algorithms can provide sound solutions for relatively simple systems, but the issue of algorithm efficiency has to be addressed for more complex systems and measures must be taken to preserve the constructive characteristic of ACS by preventing from heavy dependence on solution improvement with local searches.

The main objective of this research is to overcome the above-mentioned shortcomings and to examine the robustness and versatility of the ACS algorithm as applied to RAP for complex electromechanical systems. A further emphasis is to borrow techniques from other methods to improve ACS algorithms.

The primary contributions of this Research are in the field of stochastic reliability & redundancy allocation problems and can be listed as follows :

- A new multiobjective ACS approach is proposed to deal with RAP for a general series-parallel k-out-of-n G: subsystem. The new approach is very general in the sense that there is no limit to the component types and the component redundancy level can be from any distribution.
- The multi-objective formulation of local move strategy of ant colony is efficient and effective to construct the subsystem configuration. This approach is very useful to get good solution.
- The developed multi-objective ACS allows infeasible solutions in the search and dealing with these infeasible solutions by an adaptive penalty function. This penalty function dynamically integrates the global objective function with local search result and constraints limit.
- The elitist strategy by using long-term memory to store some ranked feasible ants enhances the good selection of components.
- Interactive approach of multi-objective ACSRAP alleviates the computational difficulties associated with the classic "a priori" methods which need preference information and restrictive assumptions, and allows more systematic, flexible, and a combined optimization tasks to be achieved.

The ACS performs very well on the redundancy apportionment problems proposed as well as on the multi-objective reliability optimization problems of gear train system, and allows us to obtain optimal design solution very frequently and more quickly than other heuristic approaches.

The remaining chapters are organized as follows. In chapter 1, a comprehensive review for RAP and solution approaches is introduced. Chapter 2 presents the background review of Ant Colony System and the mechanism of ACS algorithms. Chapter 3 presents the methodology of multiobjective ACS algorithm for a generalized RAP of series-parallel k-out-of-n : G subsystem with multiple component choice. Detailed system modeling, problem statements, ACSRAP methodology were given in this chapter. Chapter 4 represents the application of ACS algorithm for RAP. The parameter setting of ACSRAP and numerical analysis for benchmark RAP test problems are discussed to examine the effectiveness, efficiency and practicality of ACS algorithms. Chapter 5 presents the application of ACSRAP on the multiple objective reliability optimization problems for the mechanical system. The interactive ACSRAP algorithm is developed for the reliability optimization of gear train system under multi-objectives consideration. Numerical analysis for test problems are conducted and compared with those from other methods to verify the versatility and effectiveness of ACSRAP algorithm. The final conclusion summarizes the methodology of ACSRAP for reliability optimization and gives the recommendation for further works.

CHAPTER 1

RELIABILITY APPORTIONMENT PROBLEM

1.1 Problem statement

In many practical system design situation, the overall system is partitioned into a specific number of subsystems, according to the function requirement of the system. For each subsystem, there are different component types available with varying reliability, costs, weight, volume and other characteristics. The system reliability depends on the reliability of each subsystem. To maximize system reliability, the following approach can be considered : i) using more reliable component, ii) adding redundant components in parallel, or iii) a combination of i) and ii). For the systems designed using off-the-shelf components, with known cost, reliability, weight and other attributes, system reliability design can be formulated as a combinatorial optimization problem. The best-known reliability design problem of this type is the reliability & redundancy apportionment problem (RAP).

The diversity of system structures, resource constraints, and options for reliability improvement has led to the construction and analysis of several optimization models with multiple constraints, to find feasible solution for the reliability & redundancy allocation problems [1], which can then be identified as the selection of the optimal combination of component type and redundancy level for each subsystem to meet various objectives given constraints on overall system. The problem can become complicated due to the presence of multiple conflicting objectives, such as minimizing the system cost and system weight or volume, while simultaneously maximizing the system reliability. The generalized formulation of the RAP problem can be written as :

Maximize $$R_s = f(y_1, \ldots, y_s)$$

Subject to $$g_i(y_1,\ldots,y_s) \le b_i, i = 1,\ldots,m;$$
$$l_j \le y_j \le u_j, j = 1,\ldots,s;$$

where

$R_s = f(y_1,\ldots,y_s)$	System reliability
y_j	Number of components arranged in parallel at stage j, $1 < j < s$.
$g_i(\boldsymbol{y})$	Total amount of resource i required for allocation $\boldsymbol{y}$.
b_i	Constraints of resource i.
l_j	Lower limit of y_j
u_j	Upper limit of y_j

The most studied design configuration of RAP is a series system of s independent k-out-of-n: G subsystems. If l_i is more than one for all subsystems, then it is a series-parallel system which is used in our research. This is because many systems can be conceptually represented as series-parallel and because a series-parallel configuration can often serve as a base for other types of system configuration.

1.2 Approach diversities in system reliability optimization

The redundancy apportionment problem (RAP) for series-parallel systems has proven to be NP–hard [1]. Many different optimization approaches have been used for solving this problem. Tillman *et al* [2] and Kuo & Prasad [3] provided a thorough overview and summary of different formulations and different optimization approaches related to optimal system reliability with redundancy. For finite size problems, a straightforward exact algorithm is to simply enumerate the full solution space. Mathematical programming techniques, such as dynamic programming and integer programming (IP) have been successfully applied to variation of the problem.

1.2.1 Dynamic programming (DP)

DP solutions to the redundancy allocation problem are proposed in Bellman and Dreyfus (B & D) [4], Fyffe, Hines and Lee (FHL) [5], and Nakagawa and Miyazaki (N&M) [6]. The B&D and FHL formulations both use a Lagrange multiplier (λ) within the objective function to reduce the number of problem constraints to one. The formulation of the B&D and FHL algorithms becomes

Maximize $$R_s = \prod_{i=1}^{s} R_i(y_i|l_i) \exp[-\lambda \sum_{i=1}^{s} W_i(y_i)]$$

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

Where C is system cost constraint. The use of the Lagrange multiplier above may be viewed as the assignment of a penalty in the form of system unreliability to the amount of weight used and removal of the weight constraint (W) from the problem. The DP recurrence relation is as follows where the final solution is found at $f_s(C)$ for the optimal value of λ:

$$f_1(\gamma) = \max_{y_1 | C_1(y_1) \leq \gamma} \{R_1(y_1 | l_1) \exp[-\lambda W(y_1)]\}$$

$$f_i(\phi) = \max_{y_1 \left| \sum_{i=1}^{s} C_1(y_1) \leq \phi \right.} [R_i(y_i | l_i) e^{-\lambda W(y_1)} f_{i-1}(\phi - C_i(y_i))]$$

$\forall i = 2,...,s.$

The procedure of this DP algorithm can be summarized as starting with an assumed value for λ and then successively solving the recurrence formulas function $f_1(\gamma)$, $f_2(\phi)$ and continuing on until $f_{s-1}(\phi)$. Then, $f_s(C)$ is the final solution for that particular λ. If

$\sum_{i=1}^{s} W_i(x_i) = W$, then $f_s(C)$ is the optimal solution. If $\sum_{i=1}^{s} W_i(x_i) \neq W$, then λ incrementally changed, and the recurrence formulas repeated, until the weight constraint equality holds and the final solution is found. The search for the correct value of the Lagrange multiplier λ must be conducted by trail and error.

In the N&M algorithm[6], a surrogate constraint is used to combine all constraints into one, for example, $(1-u)\sum_{i=1}^{s} C_i(y_i) + u\sum_{i=1}^{s} W_i(y_i) \leq (1-u)C + uW$. An optimal solution is obtained when it satisfied the condition given by $\sum_{i=1}^{s} C_i(y_i) \leq C$ and $\sum_{i=1}^{s} W_i(y_i) \leq W$.

All examples among the published DP formulations assume l_i to be one, and only identical components can be used in parallel. This latter assumption limits the search space to subsystems where one of the available components is used exclusively. For certain constrained problems, this formulation has difficulty in identifying any feasible solution. The DP formulation with Lagrange multiplier has difficulty in solving problems with more than two constraints and inefficiency in searching for the appropriate λ. The N&M algorithm remedies some of these problems, but is inefficient compared to equivalent IP models, and cannot guarantee convergence to a feasible final solution.

1.2.2 Integer programming (IP)

IP formulations of the problem are presented by Ghare and Taylor (G&T) [7], Bulfin and Liu (B&L) [8], Misra and Sharma (MIP) [9]. The G&T algorithm presents a branch-and-bound technique to solve the redundancy allocation problem when there is only one component choice for each subsystem, and is applied to several large problems. Bulfin and Liu demonstrated that IP models can also work well when there are multiple component choices available for each subsystem. The B&L approach combines constraints into one surrogate constraint and then solves it as a knapsack problem. A typical IP formulation follows for a problem to maximize system reliability with only one component choice for each subsystem, and linearly additive cost and weight constraints :

Maximize $$\sum_{i=1}^{s} \sum_{j=k_1}^{u_{\max}} h_{ij} x_{ij}$$

Subject to $$\sum_{i=1}^{s} \sum_{j=k_1}^{u_{\max}} C_i x_{ij} \le U_1$$

$$\sum_{i=1}^{s} \sum_{j=k_1}^{u_{\max}} W_i x_{ij} \le U_2$$

where $$U_1 = C - \sum_{i=1}^{s} k_i C_i$$

$$U_2 = W - \sum_{i=1}^{s} k_i W_i$$

$$h_{ij} = \ln\left(1 - \sum_{q=0}^{k_1-1} \binom{j}{q} r_i^q (1-r_i)^{j-q} \right) - \ln\left(1 - \sum_{q=0}^{k_1-1} \binom{j}{q} r_i^q (1-r_i)^{j-q-1} \right)$$

$$x_{ij} = \begin{cases} 1, & \textit{if redundant component is used} \\ 0, & \textit{Otherwise} \end{cases}$$

r_i , C_i, and W_i represent component reliability, cost and weight for subsystem i, respectively. h_{ij} denotes the logarithm of subsystem i reliability with j components in parallel minus the logarithm of subsystem i reliability with j-1 components in parallel.

In the MIP algorithm[9], conversion of the original decision variables into binary variables is not required. This method is based on functional evaluations and limits search close to the boundary of resources. The procedure starts by computing the upper and lower bounds of all decision variables. Then, the search begins at a corner of the feasible region and finishes at another corner. The maximum permissible slack for a linear constraint i (MPS_i) is defined as a quantity slightly less than the minimum of the incremental costs among the subsystems, such as $MPS_i = \min_j (c_{ij}) - \varepsilon$. Thus, slacks are defined on the inner side of the feasible region in order to ensure that only feasible points, which are close to the boundary of the constraints, are considered for functional evaluations.

Most IP formulations assume either that $l_i = 1$ or only demonstrate examples where $l_i = 1$, and all IP formulations prohibit mixing of different components in parallel. An IP model, which considers mixing different components in parallel for each subsystem, may cause a combinatorial explosion of decision variables. Also, when the problem is to minimize cost given strict constraints on reliability and weight, IP models have difficulty in identifying feasible solutions. 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 only identical type can be used to provide redundancy. Once this restriction has been imposed, transformation 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 different characteristics. Thus, mathematical programming approaches to the problem yield "optimal solutions," but for an artificially restricted search space. Such an algorithm is infeasible due to the exponential size of the solution space.

1.2.3 Heuristic methods for RAP

To increase efficiency, all modern exact methods use pruning rules to discard parts of the search space in which the optimal solution cannot be found. These approaches are doing an implicit enumeration of the search space. For optimization problems the best-known examples are branch & bound algorithms, Generalized Lagrange Function (GLF) method and the Generalized Reduced Gradient (GRG) method.

Kuo *et al.* [10] presents a heuristic method for RAP based on a branch-and-bound and Langrangian multiplier. The initial node is associated with the relaxed version of RAP, Each node is associated with a nonlinear programming problem which is a relax version of RAP with some variables fixed at integer values. The bound associated with any node is the optimal value the corresponding optimization problem. The nonlinear programming associated with a node is solved by Langrangian multipliers.

In the nonlinear programming approach, Tillman, Hwang and Kuo [11] introduce a MINLP method to solve the RAP problem. Component reliabilities are expressed as continuous variables, cost is defined as a smooth function of reliability and the constraints are nonlinear functions of several decision variables. This approach is based on the concept that a component is added to the stage where its addition produces the greatest ratio of incremental increase in reliability to the product of decrements in slacks. This is a good approach for new system designs when components are being designed specifically for the new system and optimal reliability levels represent the reliability to be designed for the new components or used as a specified value for suppliers.

Hwang, Tillman and Kuo [12] show the Generalized Lagrange Function (GLF) method and the Generalized Reduced Gradient (GRG) method. The authors use both methods to solve nonlinear optimization problems for reliability of a complex system. They first maximize complex-system reliability with a tangent cost-function and then minimize cost with a minimum system reliability constraint. However, it is not often possible or practical to determine a differentiable function, which is required in these algorithms, for component cost as it relates to reliability.

1.2.4 Multiobjective approaches for RAP

In many practical design situations, reliability apportionment is complicated because of the presence of several (mutually) conflicting objective function. Therefore multiple objective problems (MOP) become an important aspect in the reliability design of engineering systems. Hwang et al.[13], and Lieberman [14] provide thorough article reviews related to MOP problems with mathematical programming. Hwang et al. classify multi-objective optimization problems into four categories using the timing of the information gathering process. Preference may be used before the programming process begins (a priori), progressively during the programming process (interactive), or after the programming process has been completed (a posteriori). The last category is that preference information is not used at all.

The methods with no given preference information require that the decision maker be able to accept the solution obtained from the methods. Misra and Sharma [15] used integer programming algorithm and a multi-criteria optimization method based on the min-max concept for obtaining Pareto optimal solutions of RAP. The advantage is that the decision makers will not be disturbed when making decisions; however, many assumptions about the decision maker's preferences need to be made, which is very difficult to do so.

In the "a priori" method, the decision maker needs to provide some judgment about specific objective preference level or specific trade-offs or rank the objectives in the order of their importance. Utility functions and goal programming are two of the most popular a priori methods. Rao and Dhingra [16] used goal programming formulation and goal attainment methods to generate Pareto optimal solution. If a utility functions is correctly used it will ensure the most satisfactory solution to the decision maker; however, for a complex problem with multiple objectives, it is very difficult to determine the utility function. When applying goal programming, a decision maker does not need to provide numerical weights for the multiple objectives, but needs to rank them; however, for a moderate size problem, goal programming is time consuming and needs a lot of computational capacity.

Interactive methods rely on the interaction between the decision maker and the analyst (or computer). At each iteration, a candidate solution is generated, and the decision maker is then asked about trade-off or preference information based on this solution. Based on the response, another solution which should be more attractive than the last solution is generated. This process continues until an acceptable solution is found. A multi-objective formulation of RAP to maximize system reliability and minimize the system cost has been considered by Sakawa [17] using surrogate worth tradeoff method. Inagaki, *et al* [18] used interactive optimization to design a system with minimum cost and weight. The advantage of interactive methods include that there is no need for "a priori" preference information, there are less restrictive assumptions as compared to methods described previously, and solutions obtained have a better prospect of being implemented. The disadvantages consist of no guarantee that the preferred solution can be obtained within a finite number of interactive iterations and much more effort is required from the decision maker.

A posteriori methods determine a set of non-dominated solutions and the decision maker chooses the most satisfactory solution based on some previously un-indicated preference or trade-off information. This class of methods does not require any assumption or information regarding the decision maker's preference. However, a posteriori methods usually generate a large number of non-dominated solutions, and it becomes very difficult for the decision maker to choose the one that is the most satisfactory from the Pareto optimal set.

1.2.5 Metaheuristic approaches for RAP

In recent years, meta-heuristics, which include simulated annealing (SA), genetic algorithm (GA), and tabu search (TS), have been selected and successfully applied to handle several reliability optimization problems. Genetic Algorithm (GA) has been used to solve these problems with very good results. The GA methodology is characterized by [19]

1. Encoding of solutions
2. Generation of an initial population
3. Selection of parent solution for breeding
4. Crossover breeding operator
5. Mutation operator
6. Culling of inferior solution
7. Repeat step 3 though 6 until termination criteria is met

The GA approach is flexible, can accommodate both discrete and continuous functions, and can explore a larger search space than the corresponding DP and IP formulations. GA formulations for the RAP are proposed by Painton and Campbell [20], Levitin et al. [21], and Coit and Smith [22]-[24]. Coit and Smith provide a GA to solve the system reliability problem. The authors use a penalty guided algorithm which searches over feasible and infeasible regions to identify a final, feasible optimal, or near optimal, solution. The penalty function [23] is adaptive and responds to the search history. The GA performs well on two types of problems: redundancy allocation as originally proposed by Fyffe, et al. [5], and randomly generated problems with more complex configurations. However, there is some difficulty in determining the appropriate values for the parameters and a penalty for infeasibility in GA approaches. If these values are not selected properly, a GA can rapidly converge to a local optimum or slowly converge to the global optimum.

SA is quite effective and useful in solving the complex reliability optimization problem without having any special structure, but requires much computation with many function evaluations and tests for solution feasibility. Well-designed TS can yield excellent solutions for redundancy allocation and reliability redundancy allocation problem when attribute, tabu-tenure and aspiration criteria are appropriately defined. Difficulty for TS involved in defining effective memory structures and memory-based strategies that are problem-dependent.

Our approach to this problem applies a constructive meta-heuristic, Ant Colony System (ACS), and thus gives another evidence for its versatility. The justification for using ACS is twofold: i) the redundancy apportionment problem (RAP) is a combinatorial optimization problem with a nonlinear objective function and constrains. RAP for series-parallel systems has proven to be NP–hard. It is not likely that an exact algorithm exists to solve the problem in polynomial time. This justifies using a heuristic approach for the problem. ii) Ant Colony System (ACS) is one of the adaptive meta-heuristic optimization methods. ACS is distinctly different from other meta-heuristic methods in that it is a constructive, rather than an improvement, algorithm. In addition, ACS can be turned to obtain a proper balance of exploitation and exploration during search process. Moreover, since ACS is a population-based technique that has potential for multi-objective optimization, Bi-objective version of RAP, minimizing cost and weight subject to reliability constraint during local search phase is another focus of this research.

In next chapter, we will present in details the mechanism and procedure of Ant Colony System, and discuss the previous research works on RAP with ACS algorithms.

CHAPTER 2

LITERATURE REVIEW ON ANT COLONY SYSTEM

In this chapter, a comprehensive literature review for Ant Colony System is presented first. Then a thorough review is given by concentrating on previous work on RAP with Ant Colony System. Finally, the relationship of literature and proposed research is discussed.

2.1 Background

Ant algorithms are based on the food-searching behavior of ant colonies. Real ants are capable of finding the shortest path from a food source to their nest by exploiting pheromone information without using visual cues. While walking, ants deposit pheromone on the ground, and follow, in probability, pheromone previously deposited by other ants. A way ants exploit pheromone to find a shortest path between two points is shown in Figure 1[36].

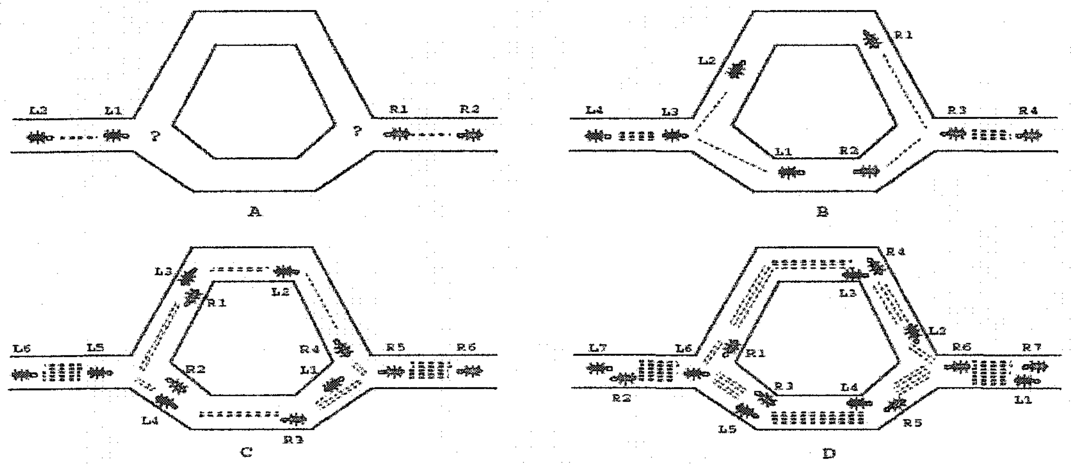


Figure 1 How real ants find a shortest path [36]

Consider Fig. 1A: Ants arrive at a decision point in which they have to decide whether to turn left or right. Since they have no clue about which is the best choice, they choose randomly. It can be expected that, on average, half of the ants decide to turn left and the other half to turn right. This happens both to ants moving from left to right (those whose name begins with an L) and to those moving from right to left (name begins with a R). Figs. 1B and 1C show what happens in the immediately following instants, supposing all ants walk at approximately the same speed. The number of dashed lines is roughly proportional to the amount of pheromone that the ants have deposited on the ground. Since the lower path is shorter than the upper one, more ants will visit it on average, and therefore pheromone accumulates faster. After a short transitory period the difference in the amount of pheromone on the two paths is sufficiently large so as to influence the decision of new ants coming into the system (this is shown by Fig. 1D). From now on, new ants will prefer in probability to choose the lower path, since at the decision point they perceive a greater amount of pheromone on the lower path. This in turn increases, with a positive feedback effect, the number of ants choosing the lower and shorter path. Very soon all ants will be using the shorter path.

The above behavior of real ants has inspired ***ant colony system*** (ACS), an algorithm in which a set of artificial ants cooperate to the solution of a problem by exchanging information via pheromone deposited on the paths. The artificial ants behave in a similar way to real ants. However, they differ in two important aspects. First, the artificial ants are not blind, i.e., they can "see" information regarding their environment and apply additional, problem-specific heuristic information; Second, they have memory, such as a tabu list in a TSP application. The characteristics of an artificial ant colony include positive feedback, negative feedback, and the use of a constructive heuristic. Positive feedback based on pheromone laying and the trail-following behavior accounts for rapid discovery of good solutions. Negative feedback implemented though pheromone evaporation avoids premature convergence of the algorithm and the constructive heuristic helps find acceptable solutions throughout the search process.

Ant system, first introduced by Dorigo et al. [25]-[26], was one of the earliest versions of ACS algorithms. Since then, ACS approach has been applied and provided solutions for various hard combinatorial optimization problems such as the structural design problems [27], telecommunication network problem [28], vehicle routing problem(VRP) [29],[38], multi-objective design [30], the traveling salesman problem (TSP) [36], the quadratic assignment problem (QAP) [37], the scheduling [39], water distribution network design [40], and continuous function problems [41], etc. Gutjahr [31] established a convergence proof for a generalized Ant System Algorithm. In an ACS algorithm, after setting the parameter values and initializing the pheromone trails, the ant colony starts to construct solutions by applying a state transition rule. Local search, if applicable, and a pheromone update rule are employed during each iteration, and the process continues until a stopping criterion is reached.

The procedure of ACS can be generalized in figure 2. According to the pheromone updating rule and state transition state updating rule, there are different ACS algorithms which are discussed subsequently.

```
Set all parameters and initialize the pheromone trails
Loop
    Sub-loop
        Construct solutions based on the state transition rule
        Apply the online update transition rule (optional)
    Continue until all ants have been generated
    Apply local search (optional)
    Evaluate all the solutions and record the best solution so far
    Apply the offline transition rule
Continue until the stop criteria is reached
```

Figure 2 Procedure of ACS algorithms [35]

2.2 State transition rule in ACS

In early Ant System (AS), solutions are constructed based on the transition probability :

$$P_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{k \in N}(\tau_{ik})^{\alpha}(\eta_{ik})^{\beta}} \tag{2.1}$$

where η_{ij} is a local heuristic, α and β are two parameters that determine the relative influence of the pheromone trail and heuristic information, and N denotes the set of candidate solutions to be chosen.

In Ant-Q algorithm, Gambardella and Dorigo [42] modified the state transition rule and the trail update rule of AS algorithm. The state transition rule is shown below :

$$y = \begin{cases} \arg\max\limits_{j \in N}[(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}] & q \leq q_0 \\ \\ Y & q > q_0 \end{cases} \tag{2.2}$$

where q is a value chosen randomly with uniform distribution in [0,1], q_0 is a parameter ($0 \leq q_0 \leq 1$), and Y is a random variable which is determine according to one of the following three rules :

- Pseudo-Random rule

  Y is selected according to the uniform distribution.

- Pseudo-Random-Proportional rule

  Y is selected according to the probability $P_{i\,j} = \dfrac{(\tau_{i\,j})^{\alpha}(\eta_{i\,j})^{\beta}}{\sum_{k \in N}(\tau_{ik})^{\alpha}(\eta_{ik})^{\beta}}$.

- Random-Proportional rule

q_0 is set to zero, i.e., all decisions are based on the probability distribution

$$P_{ij} = \frac{(\tau_{ij})^{\alpha} (\eta_{ij})^{\beta}}{\sum_{k \in N} (\tau_{ik})^{\alpha} (\eta_{ik})^{\beta}}$$

Gambardella et al. indicate that the pseudo-random-proportional rule is superior to the other two rules.

2.3 Transition update rule

In early Ant System (AS), The pheromone trail can be updated as

$$\tau_{ij}^{NC+1} = \rho \tau_{ij}^{NC} + \sum_{a=1}^{NA} \Delta \tau_{ij}^{a} \qquad (2.3)$$

where ρ is a parameter that controls the pheromone evaporated, and NA is number of ants, i.e., all ants can contribute to the pheromone trail accumulation in the AS algorithm. Dorigo et al. [36] propose three different approaches to find $\Delta \tau_{ij}^{a}$ values for the TSP as follows:

- Ant Density: $\Delta\tau_{ij}^{a} = Q$
- Ant Quantity: $\Delta \tau_{ij}^{a} = \frac{Q}{d_{ij}}$
- Ant Cycle: $\Delta \tau_{ij}^{a} = \frac{Q}{L^{a}}$

Where Q denotes a constant related the quantity of pheromone trail ants laid, d_{ij} represents the distance between cities i and j, and L^{a} is the total tour length of the

a^{th} ant. The ant density and ant quantity approaches deposit pheromone every time an ant goes from i to j, but the ant cycle deposit it only after a complete tour. Experiments indicate that ant cycle outperforms the other two approaches.

Since AS was not competitive with state-of-the-art algorithms for TSP. Researchers then proposed AS_{elite} and AS_{rank} to improve its performance. The only difference between *AS* and AS_{elite} is the pheromone update rule. In the AS algorithm, every ant has the same "weight' in contributing to the pheromone trail, but in AS_{elite}, the best ant contribute more than other ants. Therefore, the pheromone update rule is revised to

$$\tau_{ij}^{NC+1} = \rho\,\tau_{ij}^{NC} + \sum_{a=1}^{NA} \Delta\tau_{ij}^{a} + \Delta\tau_{ij}^{e} \tag{2.4}$$

where $\Delta\tau_{ij}^{e}$ can be equal to, for example, $e\frac{Q}{L^e}$ in TSP, e denoting the number of the best ants used (elitist ants), and $1/L^e$ representing the solution of the best ant found so far.

Bullnheimer at al. [43] propose an algorithm that enforces the pheromone trails by not only relaying on the elitist ants but also some other "good ants". The state transition rule is the same as the ones in AS and AS_{elite}. The contribution of an ant to the trail level update is weighted according to the rank, r, of the ant, and only the u best ants are considered, and $u = e$-1. Therefore, the trail-updating rule is as follows :

$$\tau_{ij}^{NC+1} = \rho\,\tau_{ij}^{NC} + \sum_{r=1}^{e-1} \Delta\tau_{ij}^{r} + \Delta\tau_{ij}^{e} \tag{2.5}$$

where $\Delta\tau_{ij}^{e}$, for example in TSP, is equal to $(e-r)\frac{Q}{L^r}$ and $\Delta\tau_{ij}^{e} = e\frac{Q}{L^e}$. L^e represents the solution of the best ant found so far.

In Ant-Q, after all ants complete the construction of a tour, the pheromone trail is updated by

$$\tau_{ij}^{NC+1} = \rho \tau_{ij}^{NC} + (1-\rho)[\Delta \tau_{ij}^{e} + \gamma \max \tau_{k}] \qquad (2.6)$$

where $\Delta \tau_{ij}^{e}$ uses the best solution in the iteration or the best solution found so far, and $\gamma \max \tau_{k}$ accounts for the maximum pheromone trail of the next state multiplied by a discount factor.

In order to avoid the stagnation situation in which all ants are stuck within a local optimum, Stüzle and Hoos [44] propose the MMAS algorithm to have more control on the pheromone trail. The state transition rule used is either the random-proportional rule or the pseudo-random-proportional rule. The pheromone trail is updated when all ants complete their solution construction by

$$\tau_{ij}^{NC+1} = \rho \tau_{ij}^{NC} + (1-\rho)\Delta \tau_{ij} \qquad (2.7)$$

where either the best solution in this iteration and the best solution found so far is used for $\Delta \tau_{ij}^{e}$. All τ_{ij} are initialized as τ_{max} and $\tau_{min} \leq \tau_{ij} \leq \tau_{max}$. Stüzle and Dorigo [45] also propose a variation of the state transition rule as $P_{ij} = \frac{\tau_{ij}}{\sum_{j \in N} \tau_{ij}}$ because (as shown in an MMAS application to the TSP) when local search is used to improve the algorithm, the importance of local heuristic information is replaced by local search. Therefore, local heuristic information is ignored in this version of state transition rule.

2.4 Ant Colony System (ACS) [36]

Dorigo and Gambardella propose the ACS algorithm, which is adapted from the Ant-Q algorithm. In order to balance the exploitation of good solutions and the exploration of search space, the pseudo-random-proportional rule shown as follows is used for the solution construction process:

$$y = \begin{cases} \arg \max\limits_{j \in N} [(\tau_{ij})^{\alpha} (\eta_{ij})^{\beta}] & q \le q_0 \\ \\ Y & q > q_0 \end{cases} \tag{2.8}$$

and Y is selected according to the probability $P_{ij} = \dfrac{(\tau_{ij})^{\alpha} (\eta_{ij})^{\beta}}{\sum_{k \in N} (\tau_{ik})^{\alpha} (\eta_{ik})^{\beta}}$. The α value in the Ant-Q algorithm is set to 1 in ACS because it gives the best result. Thereafter, the pheromone update rule consists of two phase-local updating (online updating) and Global Updating (offline updating). The purpose of local updating is to decay the pheromone intensity of the selected move to give more chance to exploration. Local updating is applied each time after an ant makes a move by

$$\tau_{ij}^{NC+1} = \rho \tau_{ij}^{NC} + (1 - \rho) \Delta \tau_{ij} \tag{2.9}$$

where $\Delta \tau_{ij}$ can be $\gamma \max \tau_k$, τ_0, or zero. Dorigo et al. find that the former two provide similar performance and outperform the last one. Global updating is only applied after all ants have constructed their solutions by

$$\tau_{ij}^{NC+1} = \rho \tau_{ij}^{NC} + (1 - \rho) \Delta \tau_{ij}^{e} \tag{2.10}$$

where $\Delta \tau_{ij}^{e}$ considers only the best solution found so far.

2.5 Multi-objective problem with Ant Colony System

Ant Colony System has been applied to the MOP since 1999. Mariano and Morales [40] develop an Ant-Q algorithm to a water distribution network problem which is a non-linear multiple objective optimization problem. In their algorithm, each objective is associated with a colony of ants. An objective can be determined knowing only the relevant part of a solution, and the objectives are ordered by importance. At each iteration ant *a* in colony *i* receives a solution from ant *a* of colony *i-1*, and tries to improve this solution with respect to the corresponding objective. Once the solutions have passed through all colonies, those non-dominated solutions satisfying all constraints are allowed to contribute their pheromone. Iredi, Merkle and Middendorf [46] propose an ant system problem with two objectives – minimizing total tardiness and minimizing changeover cost. Objectives cannot be ranked by importance. A heterogeneous colony approach where the ants in a colony weight the relative importance of the two objectives differently is developed. Only those ants belonging to the non-dominated set are allowed to contribute their pheromone. No online pheromone updating and no local search are used in their algorithm.

2.6 Local search for ACS algorithm

Local search plays an important role in improving the solution quality of ACS algorithms. Problem-dependent local search methods are used in different applications, such as 2-opt for the symmetric TSP, 3-opt for the asymmetric TSP [36], 2-opt and tabu search for the QAP [37], and adjacent pair-wise interchange (API) for the single machine total tardiness problem [39]. The use of a candidate list is another way to save computation time. Every time an ant makes a choice in solution construction, it first considers the candidates in the list. Thus, a reasonable length of list can decrease computation times as well as maintain good solution quality.

2.7 Heuristic information

A local heuristic η_{ij} is also a key issue to the success of the ACS algorithm. This parameter is problem-dependent, such as $\eta_{ij} = \frac{1}{d_{ij}}$ in TSP, $\eta_{ij} = \frac{1}{s_{ij}}$ in QAP, etc. d_{ij} represents the distance between cities i and j. And $s_{ij} = f_i \cdot d_j$, f_i denotes the resource flow in city i. Most ACS algorithms avoid infeasible solutions during the process of construction by means of, for example, a tabu list in the TSP application. However, Ramalhinho and Serra [47] suggest that a penalty function can be used in the objective function evaluation. For a solution violating a constraint, a penalty is added to the objective. Also, during pheromone updating, infeasible ants contribute less, as controlled by a parameter, Q.

Among variations of ACS algorithms, ACS with local search seems to be the most competitive ones because the algorithm structure is more balanced in the exploitation of good solutions and the exploration of the search space. More information of ACS algorithm can be found from the web site http://iridia.ulb.ac.be./~mdorigo/ACO/ that is maintained by Marco Dorigo.

2.8 ACS algorithm for RAP

Many researchers tried hard to solve reliability optimization problem with Ant Colony System meta-heuristic approach. Nahas & Nourelfath [32] used Ant System to solve reliability optimization for a series system. Only one component with multiple choices was used in each subsystem, and only one system constraint (system cost) was considered. Liang & Smith [33] were the first to solve the RAP for series-parallel system with a k-out-of-n: G subsystem using the Ant Colony System approach. They used duplication and mutation strategy as did in GA. Later they [34] improved their algorithm with elitist ants and local search technique. Since the results cannot compare with the solutions from Coit & Smith [22]-[24], they [35] added the following elements in their recent ACO-RAP algorithm: i). The maximum number of components in each subsystem (n_{max}) was reduced to only half of that in GARAP. ii). Each ant's solution was reconstructed with more complicated local search technique. iii). Local heuristic information and new formulation of penalty function were used. With those changes, their results for the Fyffe problem become better than those of GARAP. But since both ACO-RAP and GA are population-based algorithms, the total population number used in the algorithms, which means the total number of objective function evaluation, becomes very meaningful to compare the efficiency of the algorithms. In ACO-RAP, even with reduced search space, the maximum number of objective function evaluation in ACO-RAP increases more than 108% of that in GARAP. Therefore, the efficiency and capability of their ACO-RAP algorithm still cannot compare with GARAP algorithm since the acquisition of good results in ACO-RAP is based on the reduced search space of the problem, and also heavily depends on the improvement strategy of local search on existent solutions.

This chapter provided a comprehensive literature review for ACS algorithms, and the situation of its application for RAP. Compared with other meta-heuristic methods, ACS has the following characteristics:

- ACS is a constructive, instead of improvement algorithms like Genetic Algorithm and Tabu Search. ACS uses heuristic information during solution construction phase. This strategy should have better efficiency and effectiveness in finding good solution.
- ACS is a population-based algorithm, like GA. This characteristic provides possibility for ants to construct solution with different goals, then provides possibility for solving reliability problems with multi-objectives

As a new meta-heuristic optimization approach, ACS has demonstrated itself as a promising method for solving many NP hard combinatorial problems. But ACS algorithm hasn't yet shown its full advantage for RAP. Liang & Smith [50] provided a good orientation for the application of Ant Systems for RAP, and provides an inspiration to perform further studies of RAP with the ACS approach. This research is trying to establish a new approach based on the characteristics of ACS algorithms to solve RAP under multi-objective purpose.

Beginning with next chapter, the methodology of ACS algorithms for solving the RAP and comparison with other algorithms will be discussed thoroughly

CHAPTER 3

ANT COLONY SYSTEM ALGORITHM FOR RELIABILITY AND REDUNDANCY APPORTIONMENT PROBLEM

This chapter begins with the modeling of a generalized series-parallel systems and problem formulation of the RAP, and continues with the procedure and methodology of ACSRAP. Different versions of ACSRAP are discussed, and the differences between ACSRAP and GARAP are also compared.

3.1 Problem definition of the RAP for the series-parallel systems

3.1.1 Modeling of series-parallel system

The most studied configuration model in system design problem is the series-parallel system with k-out-of-n: G subsystem. This is because many systems can be conceptually represented as series-parallel and because a series-parallel configuration can often serve as a bound for other types of system configuration. There are many practical usage of series-parallel system. Figure 3 depicts an over-speed protection system for the gas turbine. Over-speed detection is continuously provided by the electrical and mechanical system. When an over-speed occurs, it is necessary to cut off the fuel supply by closing the five stop valves, modeled as five subsystems. To increase the reliability of each subsystem, we can use highly reliable component or/and add redundant components in parallel. Then such a system becomes a typical series-parallel system with k-out-of-n G: subsystems.

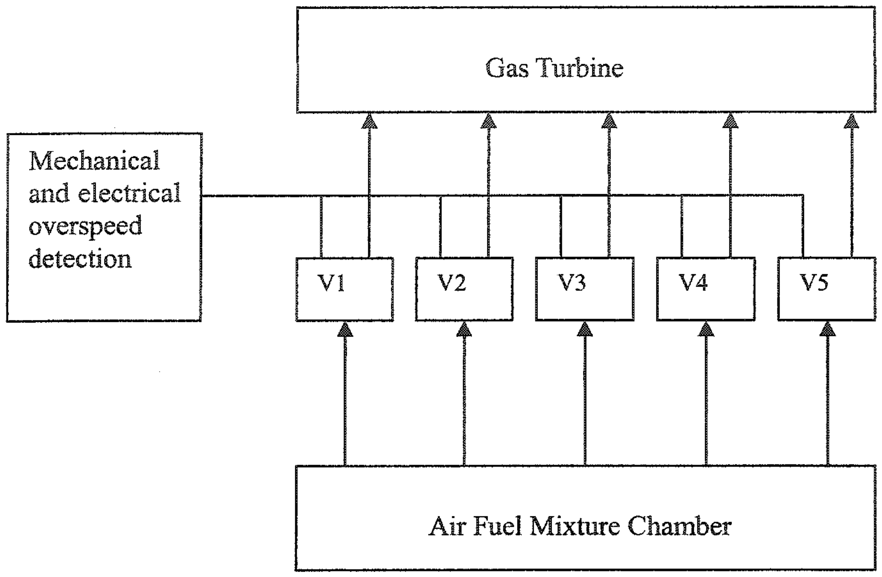


Figure 3 Schematic diagrams for the overspeed protection system of a gas turbine

3.1.2 Matrix expression of system configuration

A general series-parallel system configuration is shown in Figure 4. The system is divided into s subsystems denoted by an index i (i=1, 2…s). In subsystem i, at least p_i number of components is required for the function, and it constitutes the lower bound of the redundancy level for subsystem i. The upper bound of the component redundancy level in subsystem i is PN. The system configuration can thus be represented as a matrix of size $PN \times s$: The column index i (i =1, 2…s) represents subsystem i, and the row index k (k=1,2...PN) of the matrix represents the position where a component will be used in the subsystem. Each of the s subsystems is represented by PN positions with each component listed according to its reliability index. For the series-parallel system with k-out-of n: G subsystem redundancy problem, each possible solution is a configuration matrix of size $PN \times s$ representing a concatenation of the components in each subsystem including non-used components when $n_i \le PN$.

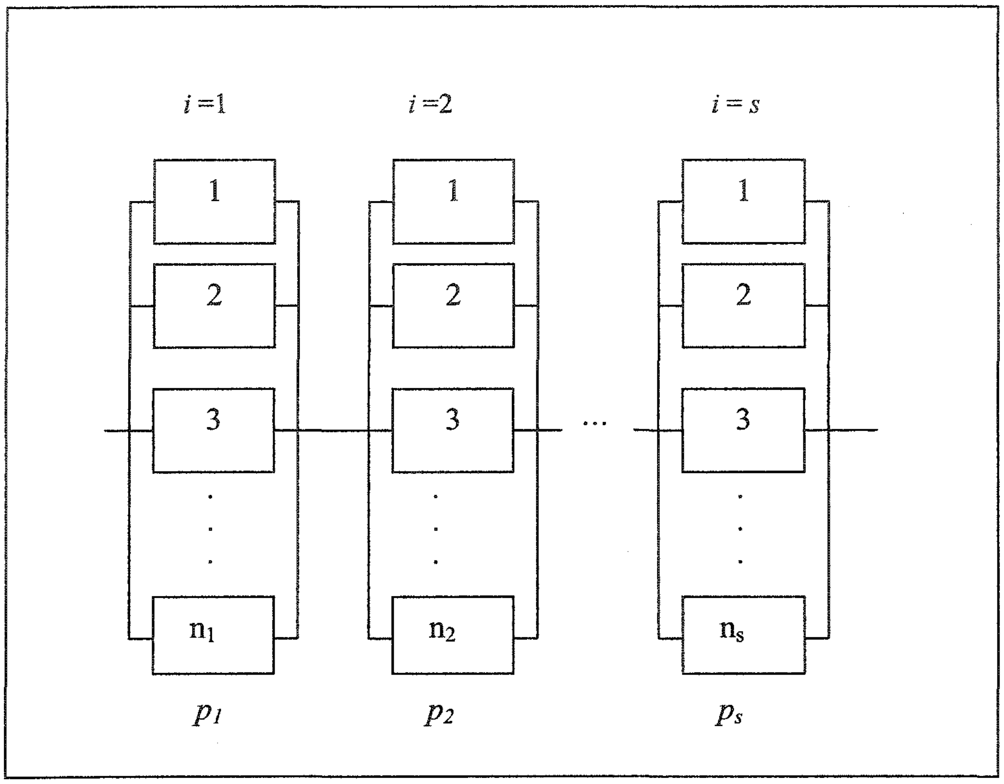


Figure 4 System configuration with k-out-of-n: G subsystem

There are m_i available component types to be chosen for subsystem i. The m_i component types are indexed in descending order according to their reliability, i.e. 1 represents the most reliable component, etc. The element x_{ki} (x_{ki} =1, 2… m_i, m_{i+1}) in the matrix of system configuration represents a component type chosen at the position k for subsystem i. An index m_{i+1} is assigned to the position where no component (defined as “blanks”) to be used in the subsystem.

For example, consider a system with s = 3, m_1 = 4, m_2 = 3, m_3 = 4 and PN = 5. The following matrix of system configuration

$$x = \begin{bmatrix} 3 & 1 & 4 \\ 3 & 2 & 4 \\ 3 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

represents a solution in which three of the third most reliable components used in parallel for the first subsystem; one of the most reliable component and two of the second most reliable components used in parallel for the second subsystem and two of the fourth most reliable components used for the third subsystem.

3.1.3 Formulation of the RAP problem for series-parallel system

Let n_i denote the total number of redundant components in subsystem i, then

$$n_i(x) = \sum_{k=1}^{PN} x_{ki} / x_{ki} \qquad x_{ki} \in (1,2,\ldots m_i) \tag{3.1}$$

From this expression, only non-blank types of components are counted. The blank component type will be counted as zero.

The reliability of subsystem $R_i(x)$ is determined by :

$$R_i(x) = 1 - \prod_{k=1}^{PN} (1 - r_{i\,x_{ki}}) \tag{3.2}$$

$$\forall i, \quad i \in \{1, 2, \cdots, s\} \text{ and } \forall x_{ki}, x_{ki} \in \{1, 2, \cdots, m_i, m_{i+1}\}$$

$$r_{i\,x_{ki}} = \begin{cases} r_{ij} & if \quad x_{ki} \in \{1,2,\cdots,m_i\} \\ 0 & if \quad x_{ki} = m_{i+1} \end{cases} \tag{3.3}$$

System reliability can be determined as:

$$R_s(\boldsymbol{x}) = \prod_{i=1}^{s} R_i(\boldsymbol{x}) \tag{3.4}$$

System cost is accumulated by the component cost in each subsystem and given by :

$$C_s(\boldsymbol{x}) = \sum_{i=1}^{s} C_i(\boldsymbol{x}) = \sum_{i=1}^{s} \sum_{k=1}^{PN} c_{i\,x_{ki}} \tag{3.5}$$

$$c_{i\,x_{ki}} = \begin{cases} c_{ij} & if \quad x_{ki} \in \{1,2,\cdots,m_i\} \\ 0 & if \quad x_{ki} = m_{i+1} \end{cases} \tag{3.6}$$

System weight is given by the sum of the component weight in each subsystem and expressed as :

$$W_s(\boldsymbol{x}) = \sum_{i=1}^{s} W_i(\boldsymbol{x}) = \sum_{i=1}^{s} \sum_{k=1}^{PN} w_{i\,x_{ki}} \tag{3.7}$$

$$w_{i\,x_{ki}} = \begin{cases} w_{ij} & if \quad x_{ki} \in \{1,2,\cdots,m_i\} \\ 0 & if \quad x_{ki} = m_{i+1} \end{cases} \tag{3.8}$$

From these expressions, the system cost and weight can be assumed as a linear function with respect to the combination of component cost and weight. But this is not a restriction for ACS. ACS can handle nonlinear constraints.

The formulation of RAP discussed here is to select how many redundant components and what type of component to use in each subsystem in order to maximize the global system reliability given the restrictions on overall system cost ($Cs_{\max}$) and weight ($Ws_{\max}$), or to minimize the system cost or weight given the requirement on system reliability (Rs_{min}) and overall system weight and cost constraints. The reliability and redundancy apportionment optimization of series-parallel system can be formulated as follows :

- Problem(P1): reliability maximization

Maximise $$R_s(x) = \prod_{i=1}^{s} R_i(x)$$

Subject to $$C_s(x) = \sum_{i=1}^{s} C_i(x) = \sum_{i=1}^{s} \sum_{k=1}^{PN} c_{i\,x_{ki}} \le Cs_{\max}$$

$$W_s(x) = \sum_{i=1}^{s} W_i(x) = \sum_{i=1}^{s} \sum_{k=1}^{PN} w_{i\,x_{ki}} \le Ws_{\max}$$

$$p_i \le n_i \le PN$$

- Problem (P2): cost minimization

Minimize $$C_s(x) = \sum_{i=1}^{s} C_i(x)$$

Subject to $$R_s(x) = \prod_{i=1}^{s} R_i(x) \ge Rs_{min}$$

$$W_s(x) = \sum_{i=1}^{s} W_i(x) = \sum_{i=1}^{s} \sum_{k=1}^{PN} w_{i\,x_{ki}} \le Ws_{\max}$$

$$C_s(x) = \sum_{i=1}^{s} C_i(x) = \sum_{i=1}^{s} \sum_{k=1}^{PN} c_{i x_{ki}} \leq Cs_{\max}$$

$$p_i \leq n_i \leq PN$$

- Problem (P3): weight minimization

Minimize $$W_s(x) = \sum_{i=1}^{s} W_i(x)$$

Subject to $$R_s(x) = \prod_{i=1}^{s} R_i(x) \quad \geq Rs_{min}$$

$$C_s(x) = \sum_{i=1}^{s} C_i(x) = \sum_{i=1}^{s} \sum_{k=1}^{PN} c_{i x_{ki}} \leq Cs_{\max}$$

$$W_s(x) = \sum_{i=1}^{s} W_i(x) = \sum_{i=1}^{s} \sum_{k=1}^{PN} w_{i x_{ki}} \leq Ws_{\max}$$

$$p_i \leq n_i \leq PN$$

Within these formulations, system weight and cost are often defined as linearly additive functions because they are reasonable representations of the cumulative component cost and weight. Assuming an upper bound on the number of redundant components (*PN*), the number of unique system configuration is given by the following equation [48]:

$$N = \prod_{i=1}^{s} \left[\binom{m_i + PN}{m_i} - \binom{m_i + p_i - 1}{m_i} \right] \tag{3.9}$$

The size of the search space is very large for the series-parallel reliability problem even though the system size is small or moderate.

3.1.4 Other assumptions regarding modeling of series-parallel system

Before discussing the ACS-RAP methodology for series-parallel system, there are the following assumptions:

- The states of components and the system have only two options - good or failed.
- Each subsystem is in series and is essential for successful operation of the system. The failure of any subsystem will cause system failure.
- If the number of good components is less than p_i in a subsystem i, then subsystem i fails.
- Failures of components are independent events.
- Failed components do not damage the subsystem, 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 including reliabilities are known and deterministic.
- The supply of components is unlimited (i.e., off-the-shelf).
- The subsystem cost is additive.

3.2 Ant Colony System for the RAP (ACSRAP)

ACS approach for RAP involves a colony of artificial ants move on each subsystem by selecting component and redundancy level. Each ant represents one configuration matrix design of a system. Ants move by applying a stochastic local decision policy that makes use of pheromone trails and heuristic information. By moving, ants incrementally build solutions to RAP. Once an ant has built a solution, or while the solution is being built, the ant evaluates the solution and deposits pheromone trails on the components it used. This pheromone information will direct the search of the future ants. The trail with more pheromone has a higher probability to be chosen by the future ants.

The Ant Colony System (ACS) algorithm for RAP proceeds as follows:

i) Initialize pheromone trail and transition probability using heuristic information.

ii) Generate ant colony and construct ant's solution by state transition rule and deterministic local move policy under multiobjective formulation.

iii) Apply local search (optional).

iv) Evaluate all solutions with dynamic penalty function and apply offline update rule to update pheromone trail and transition probability.

v) Store a number of ranked feasible ants and record the best feasible solution during iteration.

vi) Repeat steps ii) through v) until termination criteria is met.

3.2.1 Initial pheromone trails and heuristic information

An initial pheromone τ_{ij} is defined as the probability of selecting a component type j ($j = 1$, $2 \ldots m_i$) in subsystem i without any consideration of any heuristic information, and is assigned by

$$\tau_{ij} = \frac{1}{m_i} \tag{3.10}$$

For the blank choice, $\tau_{ij} = 0$. The transition of pheromone trail $\Delta\tau_{ij}$ is initialized to zero for all cases.

Since each subsystem consists of minimum p_i components, the local heuristic information is defined as the potential goodness of a component assignment in subsystem to improve the reliability of subsystem with minimum resource, and is determined by

$$\eta_{ij} = \frac{r_{ij}^{p_i}}{p_i \times (c_{ij} + w_{ij})} \tag{3.11}$$

where p_i is an integer related with the redundancy level of subsystem i and set to 1 for all subsystem. r_{ij}, c_{ij}, w_{ij} represent the associated reliability, cost and weight of component j for subsystem i. Components with higher reliability and smaller cost and weight will have greater probability to be selected.

After establishing the initial pheromone trail and heuristic information, the initial transition probability P_{ij} of component choice for ACSRAP is calculated as follows:

$$P_{ix_{ki}} = \frac{\tau_{ix_{ki}} (\eta_{ix_{ki}})^{\beta}}{\sum_{j \in N} \tau_{ij} (\eta_{ij})^{\beta}} \tag{3.12}$$

where β is a parameter that control the relative weight of the local heuristic, $\beta \geq 0$. N is the set of available component choices for subsystem i. The heuristic information η_{ij} is associated with pheromone τ_{ij} proportionally in the form of $\tau_{ij}(\eta_{ij})^{\beta}$, which becomes a priori and accumulated knowledge about the problem.

For example, a subsystem i has four component types (j = 1, 2, 3, 4). The input data are listed in table I. Initial pheromone trail, heuristic information and initial probability are calculated according to equations (3.10)-(3.12) and the results are listed in table I.

Table I

Example of component choices for subsystem i

Compnent choice (j)	Reliability (r_j)	Cost (w_j)	Weight (w_j)	τ_{ij}	η_{ij}	P_{ij}
1	0.9	1	3	0.25	0.13571	0.2257
2	0.93	1	4	0.25	0.186	0.24809
3	0.91	2	2	0.25	0.2275	0.26354
4	0.95	2	5	0.25	0.225	0.26267

3.2.2 Solution Construction

Each ant represents one design of the entire system and in an optimal solution the system reliability has to be maximized subject to system cost and weight constraints. A solution of ant colony system for RAP is constructed as follows: The component types in each subsystem are sorted in descending order according to the component reliability from the input data set. Starting from the first subsystem, ant selects a component type x_{ki} in position k of subsystem i according to the following state transition rule:

$$x_{ki} = \begin{cases} \arg \max\limits_{j \in N} [\tau_{ij} (\eta_{ij})^{\beta}] & if\ q \geq q_0 \\ Y & if\ q < q_0 \end{cases} \tag{3.13}$$

where q is a random number uniformly distributed in [0,1], q_0 is a parameter ($0 \leq q_0 \leq 1$) to be tested by the user. The parameter q_0 determines the relative importance of exploitation versus exploration: Each time when an ant in a subsystem i choose a component j to move on, it samples a random number $0 < q < 1$. If $q \geq q_0$ then the best component is chosen according to Eq. (3.13) in favor of deterministic exploitation based on heuristic information, otherwise a component is chosen according to Eq. (3.14) in favor of probabilistic exploration. Y is a component type selected according to the transition probability given by

$$P_{ix_{ik}} = \begin{cases} \dfrac{\tau_{ix_{ik}}(\eta_{ix_{ik}})^{\beta}}{\sum_{j\in N}\tau_{ij}(\eta_{ij})^{\beta}} & \text{if } x_{ki} \in N \\ 0 & Otherwise \end{cases} \tag{3.14}$$

and

$$\sum_{x_{ki}=1}^{m_i} P_{i\,x_{ki}} = 1 \tag{3.15}$$

The selection of component types uses a principle of weighted (roulette) selection : A roulette disc is divided into different area with respect to the probability of component types. For example, the probabilities of four component types in table I divide the weight disc into four areas, as shown in figure 5. A random pointer is created for component index. If the pointer is in the yellow area, the component 3 will be selected.

The above selecting procedure can be expressed with piecewise function: Let r denote a random number with uniform distribution, i.e. $r \in [0,1]$, then the process of component type x_{ij} is represented by

$$x_{ij} = \begin{cases} 1 & \text{if } \; 0 < r \le P_{i1} \\ 2 & \text{if } \; P_{i1} < r \le P_{i2} \\ \vdots & \qquad \vdots \\ m_i & \text{if } \; \sum_{j=1}^{m_i-1} P_{ij} < r \le P_{i\,m_i} \end{cases} \tag{3.16}$$

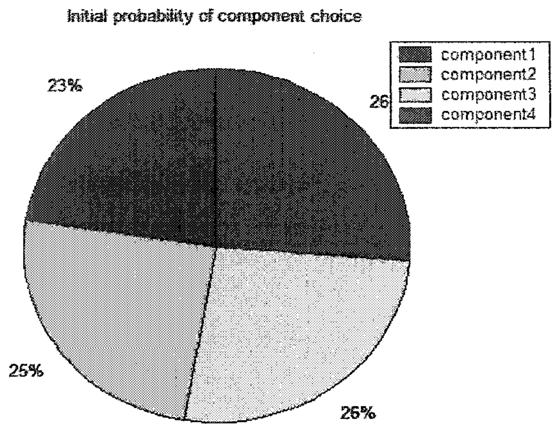


Figure 5 Initial probabilities of component types on the roulette

After ants choose a component type, they lay pheromone trail on this component, the pheromone trails and the probability of component choices will be updated after each iteration. Figure 6 shows the change of probability in figure 5 after 100 iterations. We can find out that the sum of probability for selecting component 1 and 2 is less than 1%, the probability of component 2 has 26%, and that of component 3 has 74%. Therefore, component 3 has better chance to be selected.

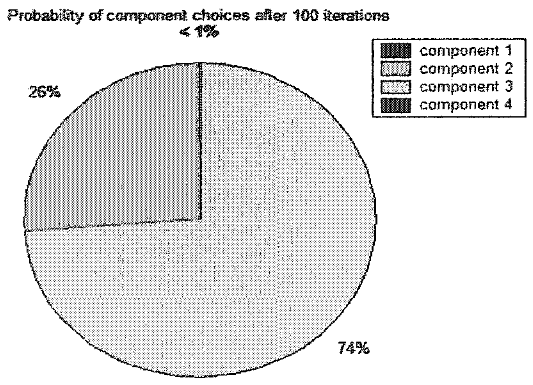


Figure 6 Probability of component choices on the roulette after 100 iterations

After ant selects components for each subsystem randomly, the reliability $R_i(x)$, weight $W_i(x)$ and cost $C_i(x)$ of each subsystem are updated and compared. An index v represents the used budget of system resource and is given by

$$v = \frac{C_s}{C_{\max}} \frac{W_{\max}}{W_s} \tag{3.17}$$

C_s, W_s are system cost and system weight respectively. C_{max} and W_{max} are the constraint limit of system cost and weight.

When a subsystem needs additional components in parallel to improve the reliability, ant uses a local move policy to select a component. The local move is under the following multi-objective formulation :

Minimize $\quad C_i(x)$ and $W_i(x)$

Subject to $\quad R_i(x|n) > \min\,[R_1(x), R_2(x), \ldots, R_s(x)]$

$C_s(x) \leq C_{\max}$, $W_s(x) \leq W_{\max}$

$n(x) = n_i + 1$

$p_i \leq n_i \leq PN$

$\forall i, i = 1,2,\ldots,s$

With this formulation, ants find the subsystem with minimum reliability and add a component type to this subsystem. If there are several subsystems that have the same minimum reliability, the subsystem is chosen according to the used resource budgets.

Example: There is a system with 14 subsystems, the constraint limit of system cost and weight are 130 and 190 respectively. During solution construction, the reliability, cost and weight of each subsystem are listed in table II. The system cost and weight are calculated and equal to 56 and 103 respectively. The budget index *v* is equal to 0.79462 according to Eq. (15). There are three subsystems (i =1, 7, 8) with the same minimum reliability that need to improve. Subsystem 1 will be chosen since it has more budgets to be used for improving the reliability.

Table II

Selection of subsystems during solution construction in ACSRAP

	Subsystem													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
R_i	0.91	0.95	0.99	0.98	0.94	0.97	0.91	0.91	0.97	0.97	0.95	0.98	0.97	0.92
C_i	2	2	4	7	2	2	4	6	2	8	4	7	2	4
W_i	2	8	11	11	4	5	7	6	8	11	6	11	6	7

After determining the subsystem to be improved, a component is assigned to the subsystem according to transition probability of components. Since redundancy and component reliability enhancement increase the system cost, a trade-off between these two options is necessary for budget-constrained reliability optimization. This can be done in two approaches: The first approach is to select a component type in the subsystem randomly. This means all the possibility of component combination will be enumerated for the subsystem configuration to reach global optimal. However, for large-scale problems with many component choices, if redundancy in a subsystem reaches a certain level, e.g. a level over four, this random search will be time consuming. To avoid such disadvantage in the ACSRAP algorithm, ant combines the probabilistic search with a deterministic approach, i.e. if redundancy reaches a certain level (user defined), a deterministic local move policy is used as follows:

- For problem (P1): reliability maximization

$$x_{i\,k+1} = \begin{cases} y = \arg \min\limits_{j \in N_i}[c_{ij}] \cap \min\limits_{j \in N_i}[w_{ij}] & \text{if } \; y \in N_i \quad \text{and} \quad v = 1 \\ \arg \min\limits_{j \in N_i}[w_{ij}] & \text{if } \; y \notin N_i \quad \text{and} \quad v \leq 1 \\ \arg \min\limits_{j \in N_i}[c_{ij}] & \text{if } \; y \notin N_i \quad \text{and} \quad v > 1 \end{cases} \tag{3.18}$$

- For problem (P2) : cost minimization

$$x_{i\,k+1} = \begin{cases} y = \arg \min\limits_{j \in N_i}[c_{ij}] \cap \max\limits_{j \in N_i}[r_{ij}] & \text{if } \; y \in N_i \\ \arg \min\limits_{j \in N_i}[c_{ij}] & \text{if } \; y \notin N_i \end{cases} \tag{3.19}$$

- For problem (P3) : weight minimization

$$x_{i\,k+1} = \begin{cases} y = \arg \min\limits_{j \in N_i}[w_{ij}] \cap \max\limits_{j \in N_i}[r_{ij}] & \text{if } \; y \in N_i \\ \arg \min\limits_{j \in N_i}[w_{ij}] & \text{if } \; y \notin N_i \end{cases} \tag{3.20}$$

with this expression, ant compares all the component choices and chooses the most suitable one to increase the reliability of the subsystem.

3.2.3 Local search

There are two types of local search strategies are used in the ACSRAP algorithm. The first local search occurs during the construction phase of the ant colony: when the redundancy in a subsystem exceeds a given level, which is determined by the system resource budget used, ants select the best component with minimum resource consumption and replace the other components in the same subsystem one by one until this subsystem uses a minimum resource budget to obtain better reliability. For example, the control level for the redundancy of subsystem i is set to 3. During the solution construction, ant configures a subsystem $x_i = [1, 2, 2, 3]^T$ with one component of type 1, two components of type 2 and one component of type 3. The redundancy level is four, and the used budget index $v < 1$. Component 3 has the least cost among components in the subsystem, then ant replaces the component 1 and 2 with component 3, and the reliability of the subsystem $R_i\,(\mathbf{x}|n)$ is recalculated until $R_i\,(x|n) > \min\,[R_1(\mathbf{x}), R_2(\mathbf{x})\ldots\ Rs(\mathbf{x})]$. The final configuration of the subsystem becomes $x_i = [3, 3, 3, 3]^T$.

The local move policy during construction phase is based on the assumption that if the redundancy of a subsystem is over a certain level, the least cost or weight components should be found to improve the reliability of the local subsystem $R_i\,(x|n)$ and improve the global reliability of the overall system. So the control level of redundancy becomes an important parameter in this strategy. When the setting of control level is small, then the deterministic local move will start early, that can speed up the solution construction but may cause the stagnation of local optima. On the other hand, if the control level is set too big, then random search will become time consuming. In next chapter, we will discuss how redundancy control influences the solution quality.

The other local search is executed after the ant colony is built in each iteration: the neighborhood of the best feasible ant is explored by a simple 2-position exchange strategy for each subsystem. Starting from the first subsystem, Components between the 1st and the last positions (non blank) are used without changing the redundancy level. The first component, if it has a lower cost and weight, replaces the last component. For example, the configuration of a subsystem in best feasible system is $x = [1, 2]^T$ with one component of type 2 and one component of type 1. Component 1 has less cost than component 2. Then ants replace the component 1 with component 2. The configuration of the subsystem becomes $x = [1, 1]^T$.

Both local search methods, the 2-position exchange and the one during construction phase in ACSRAP, do not require recalculating the entire system reliability. Every time a subsystem is changed, only the reliability of that subsystem needs to be recalculated and system reliability is updated accordingly. This process continues until system cost and weight reach constraint limit.

After the Ant Colony is constructed, the unpenalized reliability R_a of the system for each ant is calculated and sorted in descending order. The subscript a is for the a^{th} ant in a colony. Then the feasible and infeasible ants are separated in different group according to constraints limit, and sorted in descending order.

As an advantage, the solution strategy and local move policy in ACSRAP algorithm shows no sensitivity to the distribution of redundancy level for each subsystem. In other approaches such as GA, ACO-RAP [35], the redundancy level, or the number of components in parallel for each subsystem, is treated as a random number with discrete uniform distribution. For large-scale problems, this treatment causes a lot of iterations for algorithms to get optimal redundancy level of each subsystem. To limit the search space, the redundancy level in each subsystem has to be deterministically reduced in those algorithms, and then the consistence of the algorithms to the problem is distorted. In ACSRAP, the redundancy level is treated as an improvement of subsystem, which is the weakest linkage in the whole system chain. Since the ACSRAP algorithm avoids the random search of redundancy level, which is time consuming, it becomes more efficient and effective than GA and ACO-RAP.

It also should be noted that the best ant's solutions are obtained under multi-objective formulation. But the solution construction strategy for ACSRAP is different from the classical optimization methods such as goal programming or weighted-sum approaches that require user to supply a weight vector or a preference vector before solving problem. In order to obtain a set of Pareto-optimal solutions for multi-objective optimization, these methods have to be applied many times with different weight or preference vectors. Moreover, most classical approaches demand that all objectives are of the same type either all are of minimization type or of maximization type. One common way to convert a maximization problem into a minimization problem is to use the inverse function. This conversion facilitates the use of a classical optimization method, but causes a difficulty in multi-objective optimization. Such conversions do not emphasize the complete range of the transformed objective uniformly. Thus, a number of well-distributed or trade-off solutions in the original objective space may be difficult to obtain with a uniformly set of weight vectors used in the converted objective space. The solution construction strategy for ACSRAP developed here alleviates most difficulties mentioned above, thus has better flexibility to handle the multi-objective problems.

3.2.4 Penalty function

An adaptive dynamic penalty function has been successfully employed in the redundancy allocation problem with GA as developed by Coit and Smith [23]. This is incorporated into the ACSRAP algorithms as the tool of measuring the violation of constraints in order to ensure sufficient search over the feasible and infeasible regions. This approach also makes use of the ACS's property of global memory by incorporating the best solutions found so far (both feasible and infeasible) into the penalty function. The penalty function employs the notion of a "Near-Feasibility Threshold" (NFT) for each constraint. The NFT is the threshold distance from the feasible region that is considered as being close to feasibility.

Since the redundancy allocation problem is formulated with an objective function and two independent constraints, the penalty function is a linear summation as follows:

- For problem (P1): reliability maximization

$$R_{ap} = R_{as} - (R_{all} - R_{best}) \times ((\frac{\Delta W_s}{NFT_w})^{\gamma_1} + (\frac{\Delta C_s}{NFT_c})^{\gamma_2}) \tag{3.21}$$

R_{as} denotes the unpenalized system reliability of ant's solution. R_{ap} is the penalized objective function value of ant's solution. R_{all} denotes the unpenalized solution value of the best solution found; R_{best} is the value of the best feasible solution found. ΔW_s and ΔC_s represent the magnitude of the cost and weight constraints violations for ant's solution. γ_1 and γ_2 are amplification parameters and are set to two in this paper. If R_{all} and R_{best} are equal or similar in value, then the search continues essentially as an unconstrained search because feasible solutions are being found. Alternatively, if R_{all} is much larger than R_{best}, 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): cost minimization

$$C_{ap} = C_{as} + (C_{best} - C_{all}) \times ((\frac{\Delta R_s}{NFT_R})^{\gamma_1} + (\frac{\Delta C_s}{NFT_c})^{\gamma_2}) \tag{3.22}$$

C_{as} denotes the unpenalized system reliability of ant's solution. C_{ap} is the penalized objective function value of ant's solution. C_{all} denotes the unpenalized solution value of the best solution found; C_{best} is the value of the best feasible solution found. ΔR_s and ΔC_s represent the magnitude of the reliability and weight constraints violations for ant's solution.

- For problem (P3): weight minimization

$$W_{ap} = W_{as} + (W_{best} - W_{all}) \times ((\frac{\Delta R_s}{NFT_R})^{\gamma_1} + (\frac{\Delta W_s}{NFT_w})^{\gamma_2}) \tag{3.23}$$

W_{as} denotes the unpenalized system reliability of ant's solution. W_{ap} is the penalized objective function value of ant's solution. W_{all} denotes the unpenalized solution value of the best solution found; W_{best} is the value of the best feasible solution found. ΔR_s and ΔW_s represent the magnitude of the reliability and weight constraints violations for ant's solution.

NFT_c, NFT_w and NFT_R are the "feasible thresholds" and given by:

$$NFT_w = \frac{NFT_{w\max}}{1 + \lambda \times NC} \tag{3.24}$$

$$NFT_c = \frac{NFT_{c\max}}{1 + \lambda \times NC} \tag{3.25}$$

$$NFT_R = \frac{NFT_{R\min}}{1 + \lambda \times NC} \tag{3.26}$$

NC is the number of iterations, and λ is a parameter that assures the entire region between constraint limit and zeros being searched. NFT_{cmax} and NFT_{wmax} are the maximum cost and weight among the feasible solutions in each iteration. NFT_{Rmin} is the minimum reliability among the feasible solutions in each iteration

3.2.5 Pheromone update

The pheromone trail update applied every time after all ants have built solutions. In Ant Colony System, the pheromone trail is updated according to:

$$\tau_{ij}^{NC+1} = \rho \tau_{ij}^{NC} + (1-\rho)\,(\sum_{a=1}^{NA} \Delta\tau_{ij}^{a} + \sum_{f=1}^{NA_{feas}} \Delta\tau_{ij}^{f}) \tag{3.27}$$

where ρ is the persistence of pheromone trail, so that (1-ρ) represents the evaporation of pheromone from previous iteration. The parameter ρ is used to avoid unlimited accumulation of the pheromone trails and allows the algorithm to forget previously done bad choices of component type. *NA* is the quantity of ants in each ant colony. NA_{feas} denote the number of feasible ants in each iteration. $\Delta\tau_{ij}^{a}$, called as local update rule, is the amount of pheromone the a^{th} ant putting on the j^{th} component type in subsystem i and is given as follows:

- For problem (P1): reliability maximization

$$\Delta\tau_{ij}^{a} = \begin{cases} Q * R_{ap} & if\,(i,j) \in combinations\ \ used\ by\ the\ a^{th} ant\ and\ R_{ap} > 0 \\ 0 & otherwise \end{cases} \tag{3.28}$$

- For problem (P2): cost minimization

$$\Delta\tau_{ij}^{a} = \begin{cases} Q * C_{ap} & if\,(i,j) \in combinations\ \ used\ by\ the\ a^{th} ant\ and\ C_{ap} > 0 \\ 0 & otherwise \end{cases} \tag{3.29}$$

- For problem (P3): weight minimization

$$\Delta\tau_{ij}^{a} = \begin{cases} Q * W_{ap} & if\,(i,j) \in combinations\;\; used\; by\; the\; a^{th} ant\; and\; W_{ap} > 0 \\ 0 & otherwise \end{cases} \tag{3.30}$$

$\Delta\tau_{ij}^{f}$ represents the trails laid by feasible ants, and is given as follows:

- For problem (P1): reliability maximization

$$\Delta\tau_{ij}^{f} = Q * R_{best} \tag{3.31}$$

- For problem (P2): cost minimization

$$\Delta\tau_{ij}^{f} = Q * C_{best} \tag{3.32}$$

- For problem (P3): weight minimization

$$\Delta\tau_{ij}^{f} = Q * W_{best} \tag{3.33}$$

Eq. (3.31) - Eq. (3.33) are also called as global update rule. Q is the amount of pheromone trail deposited by an ant.

After updating pheromone trail, the transition probability is then updated by Eq. (3.14). Future ants choose component combination according to the updated transition probability. This process continues until the termination criteria or fixed iteration number is reached.

3.2.6. Memory of ACSRAP

One of the most characteristics of ACS is the usage of memory, which helps to enhance the pheromone trails of good solution and to prevent stagnancy of ants in local optima.

There are two kinds of memory in ACS-RAP. The first memory is short-term memory to store the pheromone change of $\Delta\tau_{ij}^{a}$ and $\Delta\tau_{ij}^{f}$ in the colony. This memory is refreshed after each iteration. The second memory is a dynamic long-term memory used for storing a number of ranked feasible ants to enhance the pheromone trail of good combination. The memory size is controlled with respect to the size of ant colony. For example, with a size of 100 ants in a colony, the long term memory for ranked feasible ants can be controlled from 20 to 30. The ranked feasible ants in this memory are updated after each iteration with the principle of first-in and first-out. The best solution is recorded and printed in each iteration.

3.3 Comparison of different versions of ACS algorithms for RAP

Formally, the ACSRAP algorithm discussed above can be defined in Figure 7. The differences between ACO-RAP[35] and ACS-RAP can be listed as follows:

- ACS-RAP deems redundancy level in each subsystem as a need for improving the weakest linkage of system chain, not as a random number as did in ASRAP.
- Local search mainly happens in construction phase of ant colony in ACS-RAP under multiple objective formulations. This strategy facilitates the ant colony to search good solution efficiently and effectively.
- Pheromone trail updating in ACS-RAP has two parts: local updating rule $\Delta\tau_{ij}^{a}$ and global updating rule $\Delta\tau_{ij}^{f}$. The former is helpful to diversify the search space and to prevent the search stuck in local optima. The latter is used to enhance

possibility of good solution for ants.

In ACS-RAP, the dynamic memory used for long–term and short-term storage of pheromone trail. Such strategy is helpful to balance the search between exploration and exploitation and prevent the search from stagnating in a local optimum. In next chapter, we will see how such strategy influences the algorithm.

```
Set all parameters and initialize the pheromone trails
Loop
    Sub-loop
        Construct solutions based on the state transition rule:
        $P_{ix_{ki}} = \frac{\tau_{ix_{ki}} (\eta_{ix_{ki}})^{\beta}}{\sum_{j=1}^{m_i} \tau_{ij} (\eta_{ij})^{\beta}}, i = 1, 2, \cdots, s$, and local move policy
        under multiple objective formulation
        Continue until all ants in the colony have been generated
    Apply 2-position Opt local search to best ant (optional)
    Evaluate all the solutions and record the best solution so far
    Apply offline transition rule:
    $\tau_{ij}^{NC+1} = \rho \tau_{ij}^{NC} + (1 - \rho)(\sum_{a=1}^{NA} \Delta \tau_{ij}^{a} + \sum_{f=1}^{NA_{feas}} \Delta \tau_{ij}^{f})$
Continue until the stop criteria is reached
```

Figure 7 Procedure of ACSRAP algorithm

3.4 Comparison of ACSRAP algorithm with GARAP

The basic schema of GARAP is shown in figure 8. Both ACSRAP and GARAP are probabilistic and population based algorithm. But if we compare their procedure and their methodology, we can find their differences as follows:

- GARAP initializes the generation of the solutions randomly, and then improves them through several operation procedures. Therefore GARAP is an improvement strategy. But ACS-RAP initializes the solutions of Ant Colony through a combination of random search, local move policy, and local heuristic information. Ants are very greedy to find good solutions during construction phase. Therefore ACSRAP is a constructive strategy.
- In GARAP, both component type and redundancy level are treated as random number. Too much random search influences the efficiency and consistency of the algorithm. In ACSRAP, such problems are alleviated, and then a better efficiency can be expected.

```
Set all parameters and encoding of solutions
Generate an initial population
Loop
    Selection of parent solution for breeding
    Crossover breeding operator
    Mutation operator
    Culling of inferior solution
Continue until the stop criteria is reached
```

Figure 8 Schema block of GARAP

This chapter discussed the methodology and the procedure of ACSRAP. In next chapter, we will verify the efficiency and effectiveness of ACSRAP with benchmark problems.

CHAPTER 4

APPLICATION OF ACSRAP ALGORITHM FOR RAP BENCHMARK PROBLEMS

In this chapter, benchmark problems for RAP will be tested with ACS to demonstrate the efficiency and effectiveness of ACSRAP algorithm. The influence of parameters of ACSRAP to the solutions will be discussed.

4.1 Test Problems

The best known problem of a series-parallel system was originally proposed by Fyffe *et al* [5], they specified a k-out-of-n subsystems with 130 units of system cost, 170 units of system weight and k_i=1; i.e., 1-out-of-n:G subsystems. In the optimal solution, the weight constraint was active and the cost of system was 119. Nakagawa and Miyazaki [6] devised 33 variations of the original problem. They fixed the cost constraint C_{max} = 130 and the weight constraint varies from 159 to 191. The formulation of the problems is the reliability maximization problem (P1) as did in chapter 3 to maximize the global system reliability given the restrictions on overall system cost and weight. In both papers, the approach required that only identical components could be placed in redundancy. Coit and Smith [22]-[24] solved this problem with a GA without restricting the component mixing. The maximum redundancy level in subsystem PN =8 and the search space is larger than 7.6×10^{33}. Liang & Smith [35] used ACO-RAP solved the same problem. They restricted the maximum redundancy level to 4 and the minimum redundancy level p_i = 2. The search space in ACORAP is less than 6.45×10^{22}.

In this paper, It is assumed that the minimum redundancy level p_i =1 and PN = 10 for all subsystems. Considering component mixing, the size of search space for unique system configuration is larger than 4.3×10^{37}. The input data of components characteristics for this problem is summarized in Table III.

Table III

Input data for RAP benchmark problem

	Component type											
	1			2			3			4		
sub-system	r1	c1	w1	r2	c2	w2	r3	c3	w3	r4	c4	w4
1	0.90	1	3	0.93	1	4	0.91	2	2	0.95	2	5
2	0.95	2	8	0.94	1	10	0.93	1	9	*	*	*
3	0.85	2	7	0.90	3	5	0.87	1	6	0.92	4	4
4	0.83	3	5	0.87	4	6	0.85	5	4	*	*	*
5	0.94	2	4	0.93	2	3	0.95	3	5	*	*	*
6	0.99	3	5	0.98	3	4	0.97	2	5	0.96	2	4
7	0.91	4	7	0.92	4	8	0.94	5	9	*	*	*
8	0.81	3	4	0.90	5	7	0.91	6	6	*	*	*
9	0.97	2	8	0.99	3	9	0.96	4	7	0.91	3	8
10	0.83	4	6	0.85	4	5	0.90	5	6	*	*	*
11	0.94	3	5	0.95	4	6	0.96	5	6	*	*	*
12	0.79	2	4	0.82	3	5	0.85	4	6	0.90	5	7
13	0.98	2	5	0.99	3	5	0.97	2	6	*	*	*
14	0.90	4	6	0.92	4	7	0.95	5	6	0.99	6	9

All ACSRAP algorithms are coded in Matlab and tests of ACSRAP algorithm are running with an Intel Pentium IV 2.2 GH_z PC with 256MB 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 literature.

4.2 Parameter settings for ACSRAP

The parameters setting of the ACSRAP algorithm are determined by trial and error according to the results and computing time. The default parameter settings of ACSRAP are set to the following values except as indicated differently :

- Number of ants for every iteration $NA = 100$
- Memory size for global update rule of pheromone trail $u = 20$
- Number of iteration $NC_{max} = 300$.
- Relative importance of the heuristic information $\beta = 0.3$
- The control level of redundancy for each subsystem is set to 1
- Constant to assure that the entire searching region between NFT and zero $\lambda = 0.4$
- Preset severity parameter for penalty $\gamma_1 = \gamma_2 = 2$
- Trail persistence $\rho = 0.6$
- Constant for pheromone updating $Q = 1$
- Relative influence of exploitation versus exploration $q_0 = 0.9$
- Cost constraints $Cs_{max} = 130$
- Weight constraints Ws_{max} is changed from 191 to 159

The stopping criteria of ACO-RAP are either when the total number of iterations reaches 300 or the best ant has not changed for 100 consecutive iterations. Each instance is run 10 times according to different random number seeds and then the maximum and standard deviation of the best ants over 10 runs are chosen for comparison.

In order to investigate the influence of parameter setting of ACSRAP algorithm to the solutions of the problems, the following parameters are changed :

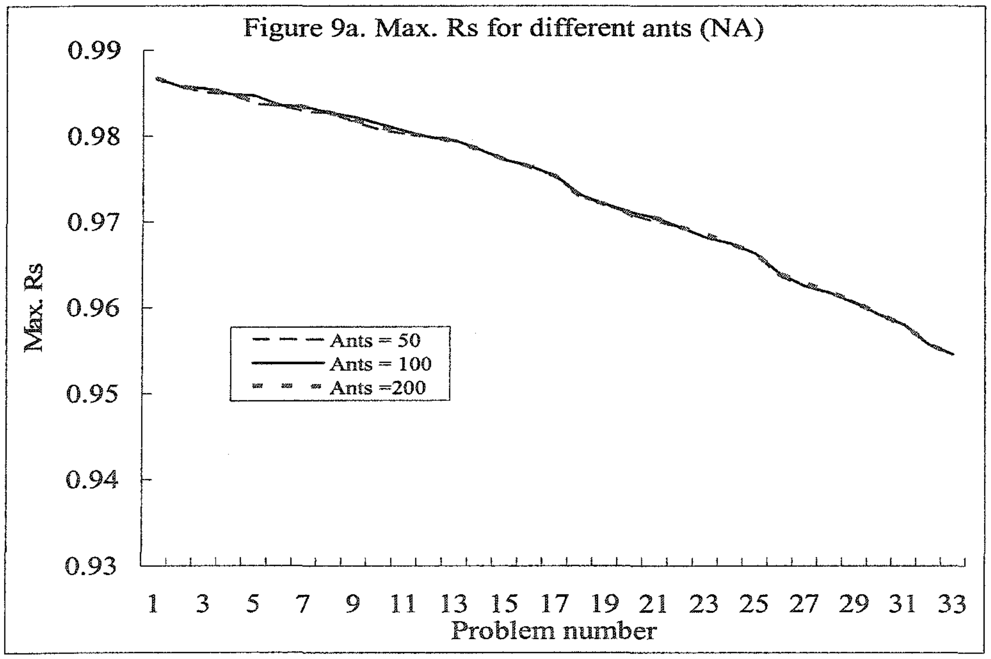
- Number of ants NA for each iteration
- Relative importance of the heuristic information β

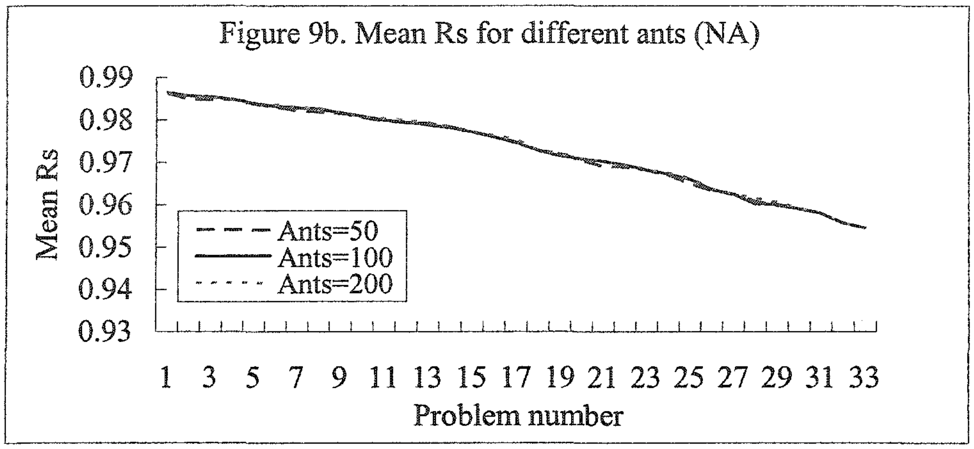
- Memory size for global update rule of pheromone trail u
- Amplification constant for pheromone updating Q
- Trail persistence ρ

4.2.1 Number of ants

In general, the best number of ants is a function of the particular ACS algorithm chosen as well as of the class of problems being attacked, and most of the times it must be set experimentally. Fortunately, ACS algorithms seem to be rather robust to the actual number of ants used.

Figure 9a – 9c shows the results of 33 problems with 50, 100, 200 ants with 10 runs. The memory size of ranked best feasible ants is set to 10, 20 and 40 respectively.





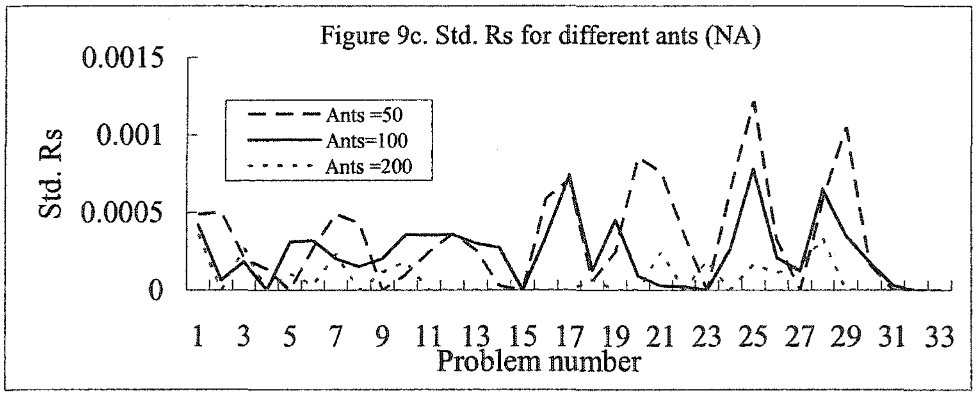


Figure 9 Influence of ant number on the best feasible solution in ACSRAP

It is found out that the standard deviation of *Rs* decreases with respect to the number of ants. However, increasing number of ants means increasing computation time. From the maximum and mean value of *Rs*, the results of 100 ants are better than the ones with 50, but do not show much difference with 200 ants. Therefore, the size of ant colony is set to 100 for the rest discussion.

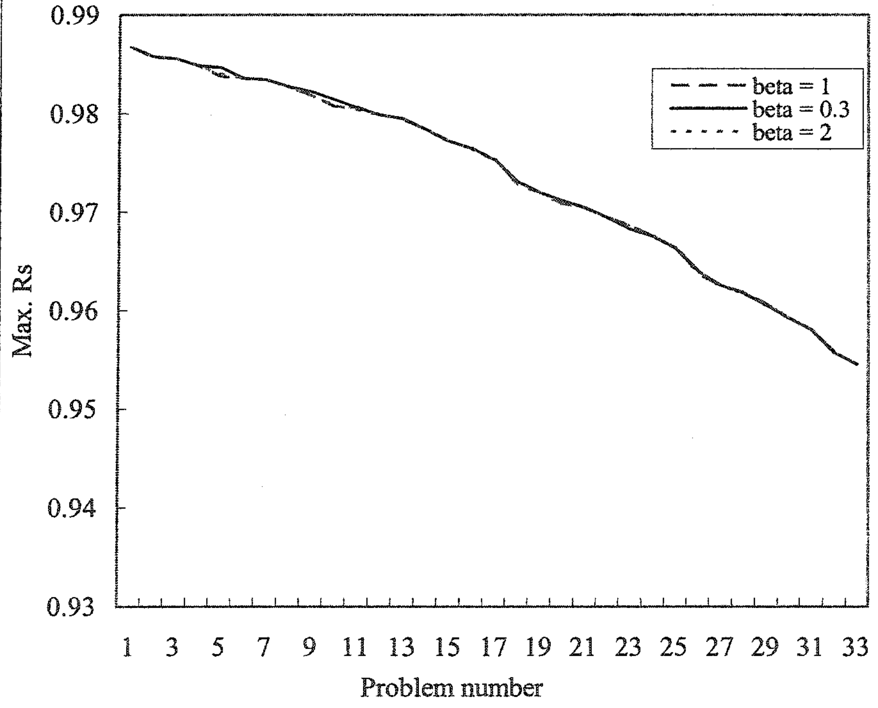
4.2.2 Importance of heuristic information β

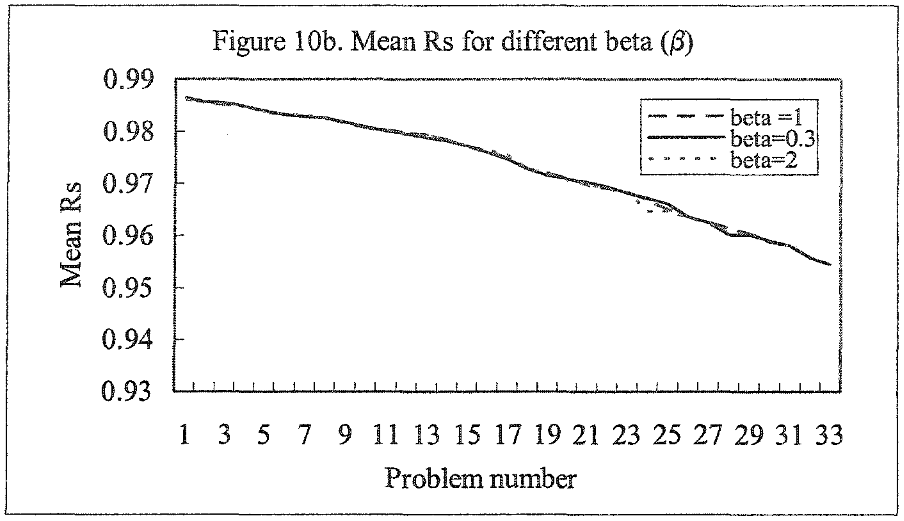
In order to investigate the relative importance of the problem specific heuristic in the

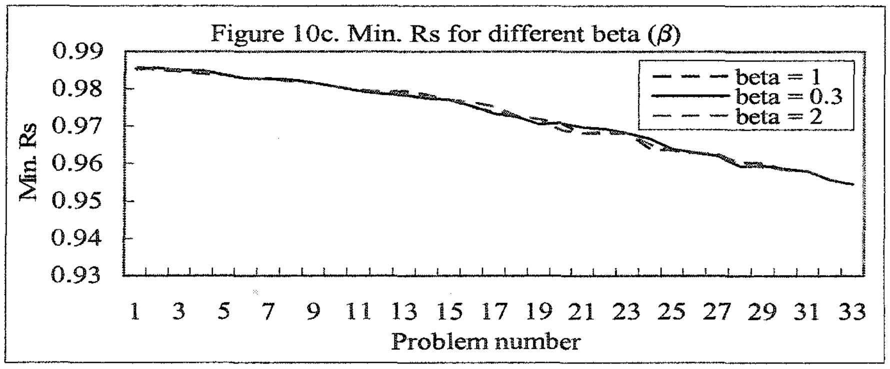
ACS-RAP algorithm, experiments are conducted using different β values that control the relative weight of the local heuristic. The ant number is set to 100.

Figure 10 shows the system reliability of the best feasible solution over 10 runs for each of the 33 instances. When $\beta = 0.3$, ACS-RAP is even or superior to $\beta = 1$ and $\beta = 2$ with very close variability, and $\beta = 2$ shows smaller standard deviations in most cases. In those cases where $\beta = 2$ outperforms $\beta = 0.3$ and $\beta = 1$ in terms of the standard deviation measure, the $\beta = 2$ version is stuck in a local optimum, especially for high constraint problems, which leads to the smaller variation. Therefore, the standard deviation should be seen as a secondary measure of performance in this case, while the maximum of the best feasible solutions over 10 runs is regarded as the main measurement of performance. For the rest of experiments, β is fixed to 0.3.

Figure 10a. Max. Rs for different beta (β)







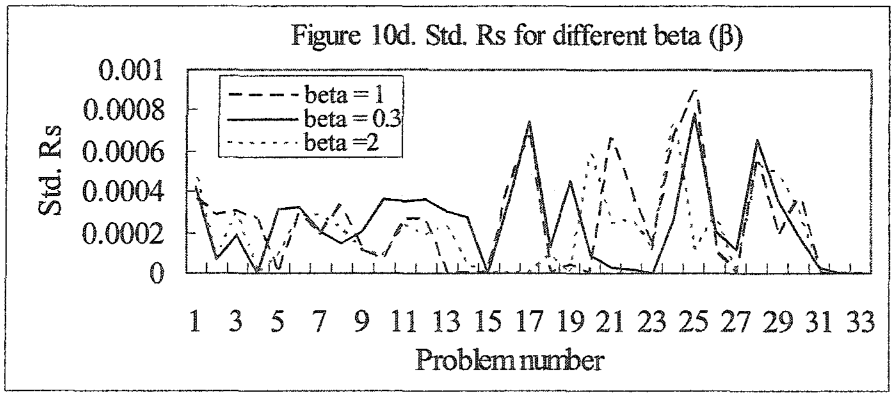
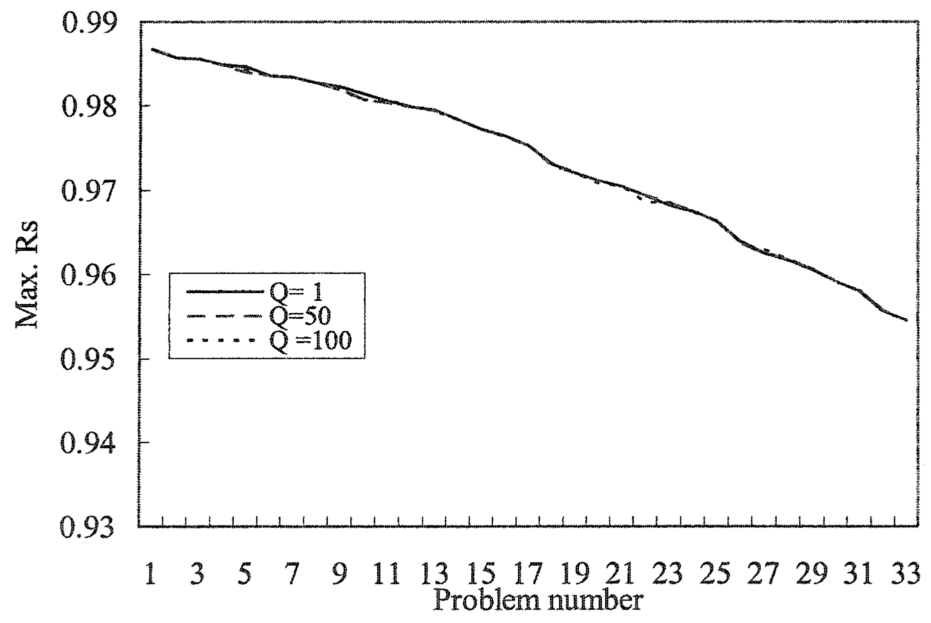


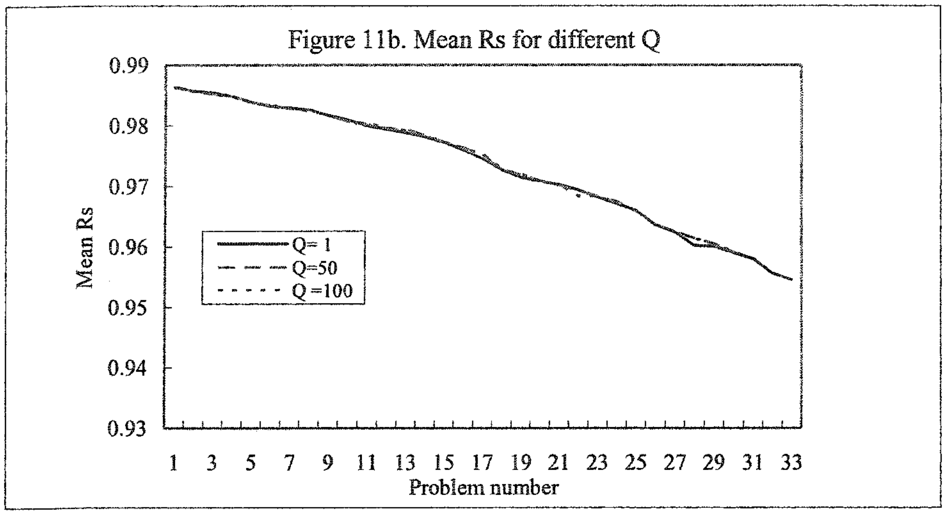
Figure 10 Influence of heuristic information (β) in ACSRAP

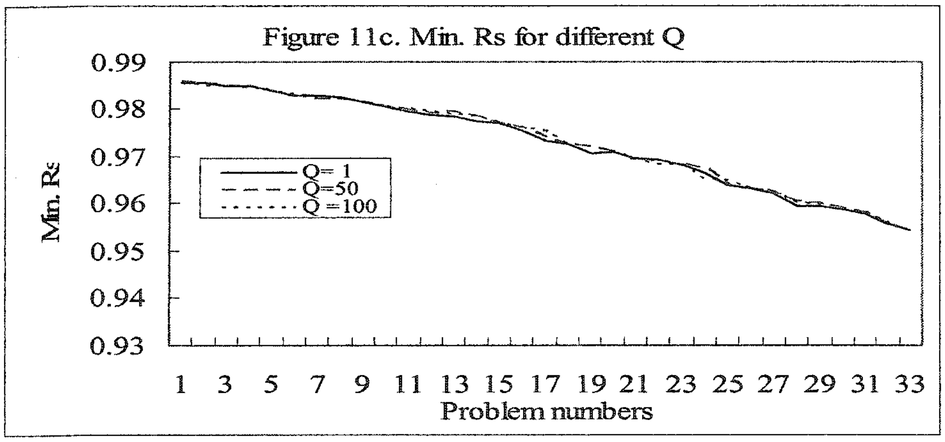
4.2.3 Amplification constant of pheromone trail *Q*

Three different values of *Q* (1, 50, 100) are tested to demonstrate the influence of amplification factor of pheromone trail. The results in figure 11 show that there are very similar results in terms of maximum system reliability although Q = 100 give smaller standard deviation than Q = 1 and *Q* = 50. Since Q =1 outperforms Q =50 and Q=100 in terms of maximum system reliability over 10 runs for most cases of 33 variations. Therefore, we set Q =1 for the rest of discussion.

Figure 11a. Max. Rs for different Q







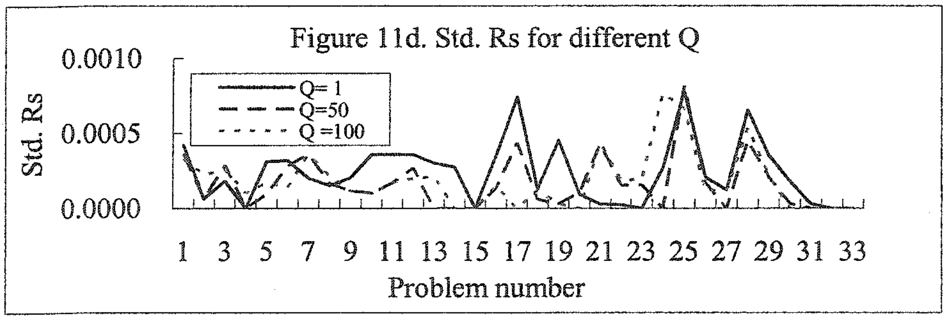
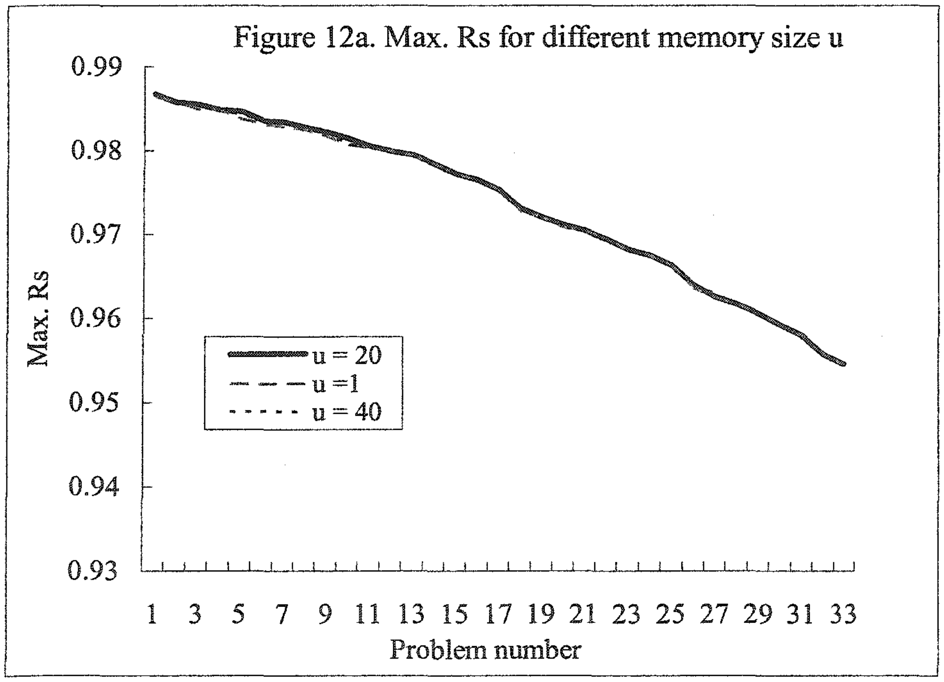


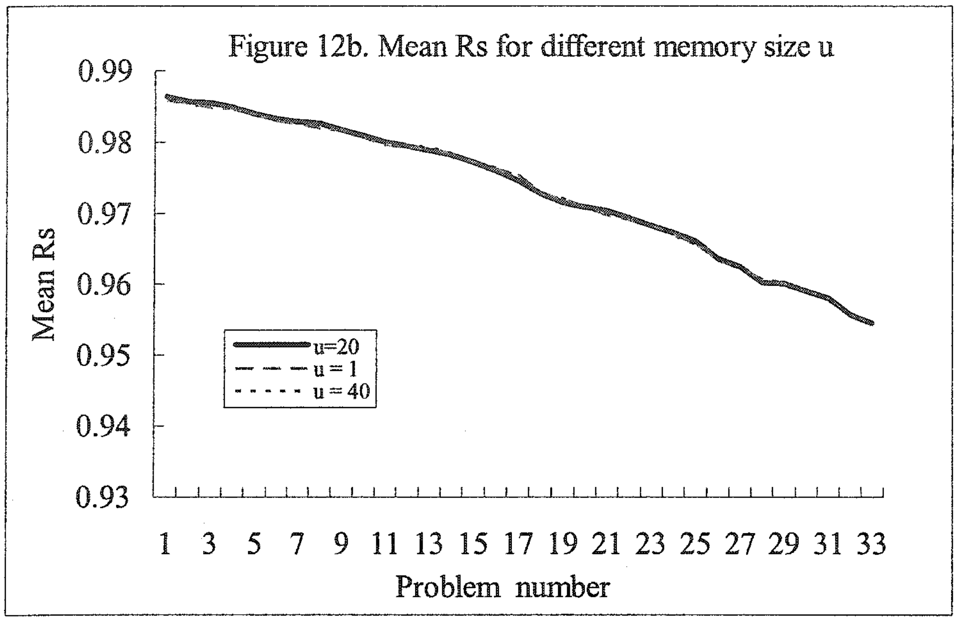
Figure 11 Influence of amplification factor of pheromone trail Q

4.2.4 The role of memory size (u) for global update rule of pheromone trail

In order to examine the influence of dynamic long term memory of the ranked feasible ants for pheromone updating, experiments are conducted. Three different memory sizes (u = 1, 20, 40) are tested respectively in the ACS-RAP algorithm. The ranked version allows the top u ranked ants stored in the long-term memory at each iteration and the best feasible ants to contribute pheromone. If the memory size u =1, there will be no ranked ants to deposit the pheromone, only the globally best feasible ant to deposit the pheromone.

The system reliability of the best feasible solution over 10 runs for each of the 33 instances is shown in figure 12. From the maximum reliability, the performance of the ranked version is better than no memory version in all 33 variations of problem. These "good" ants do help balance the search between exploration and exploitation and successfully prevent the search from stagnating in a local optimum.





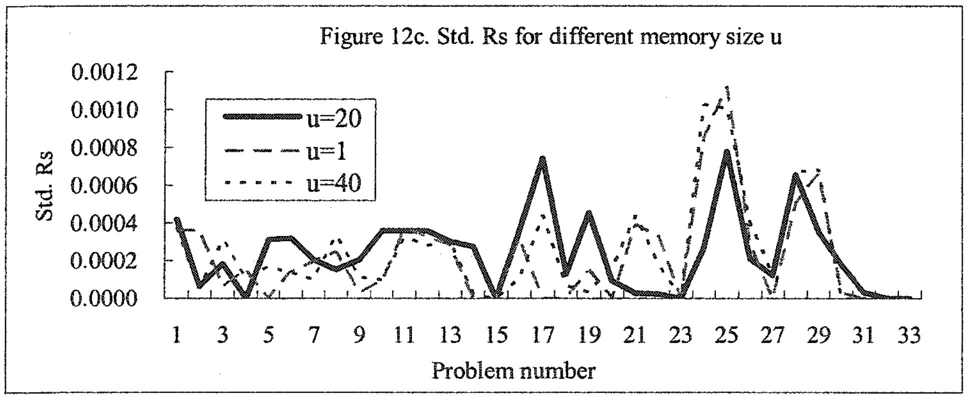


Figure 12 Influence of long term memory size for ranked feasible ants in ACSRAP

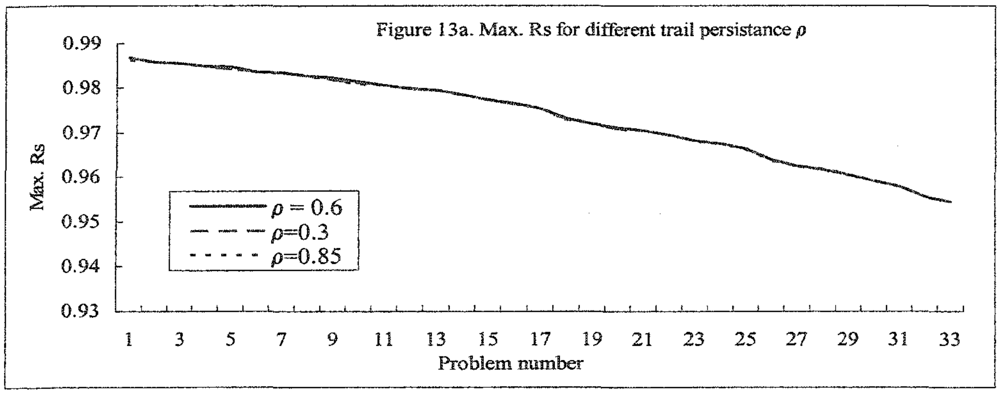
The standard deviation of the best feasible solution to each of the 33 instances is shown in figure 12c. The performance of memory strategies on variability is very close. But the medium size of memory shows better results in terms of maximum reliability. In those cases where the ranked version have low performance in terms of the standard deviation measure, the size of memory is sensitive to the maximum reliability, too big (u = 40) or too small size (u=1) of memory is stuck in a local optimum which leads to the smaller variation. Therefore, the standard deviation should be seen as a secondary measure of performance in this case, while the maximum of the best feasible solution over 10 runs is regarded as the most important measurement of performance.

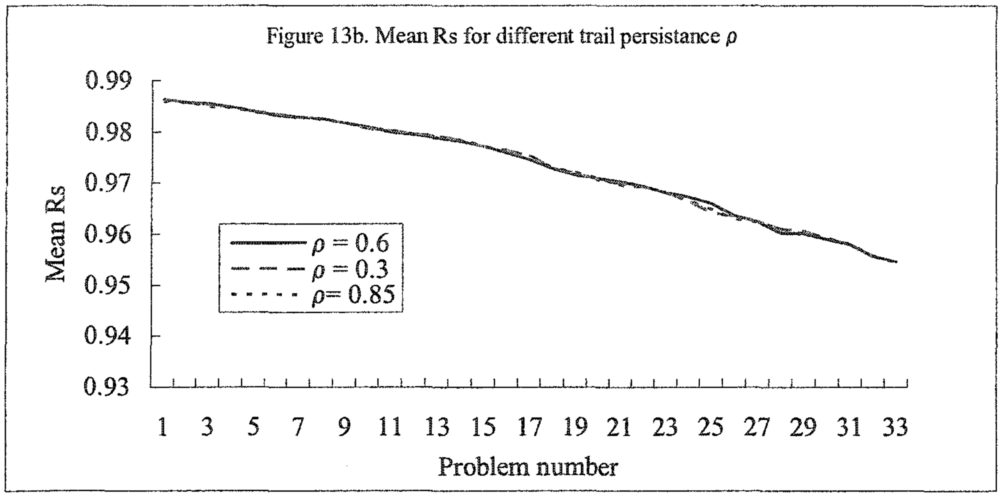
Considering overall performance from figure 12a to figure 12c, the memory size of $u = 20$ outperforms the others. Further test shows that the memory size from 20 to 30 has good results.

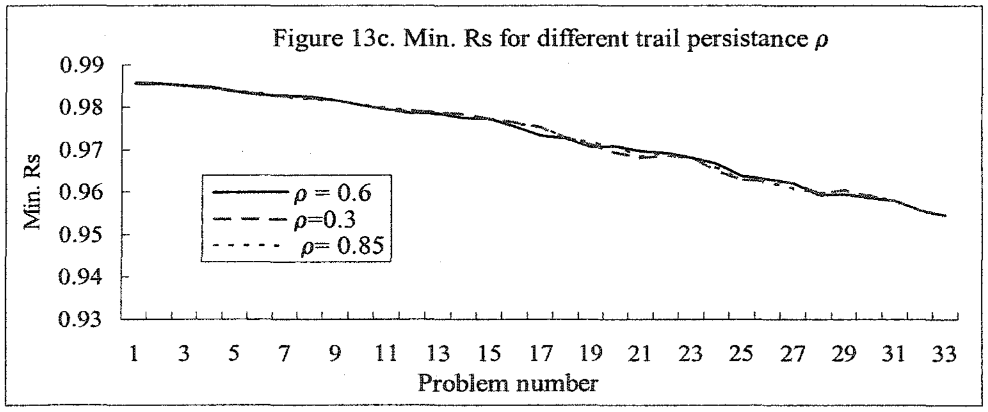
4.2.5 Influence of pheromone trail persistence ρ

The parameter ρ influences the evaporation of pheromone trail laid by ant colony during solution construction. To exam the influence of trail persistence ρ, experiments are conducted with different values of ρ ($\rho = 0.3, 0.6, 0.85$). The results are shown in figure 13 .

When $\rho = 0.6$, ACSRAP is even or superior to the others in measure of maximum system reliability with very close variability. For the medium constraint problems, $\rho = 0.85$ and 0.3 outperform $\rho = 0.85$ in measure of min. reliability. These results indicate that too high or too low evaporation of pheromone trail leads a local optimum more frequently. Therefore, pheromone persistence ρ is set to 0.6 in the rest of experiment.







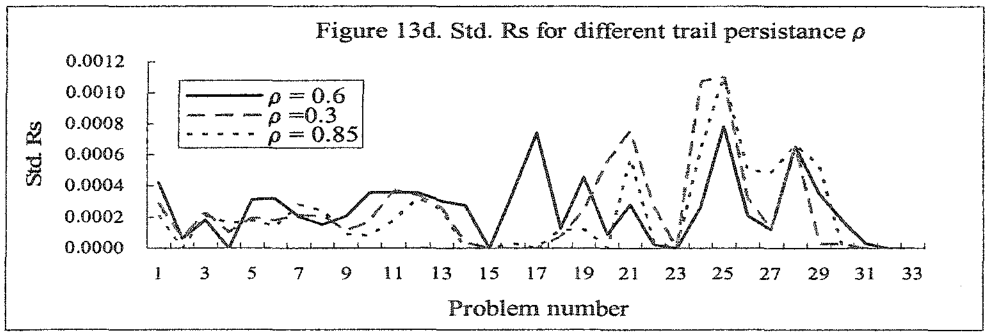


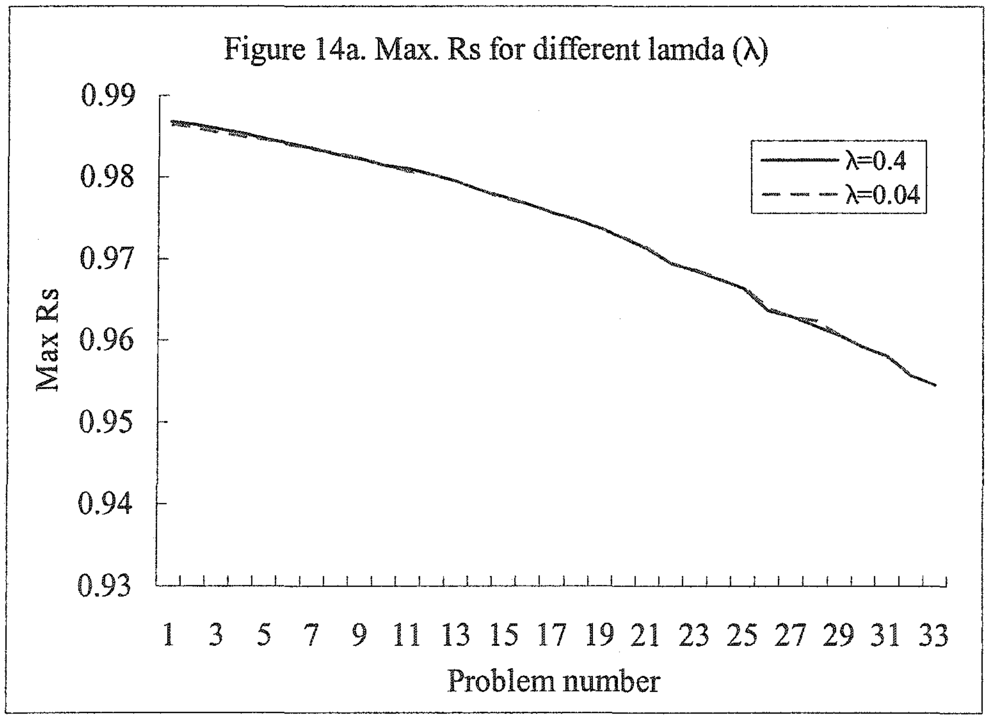
Figure 13 Influence of pheromone trail persistence

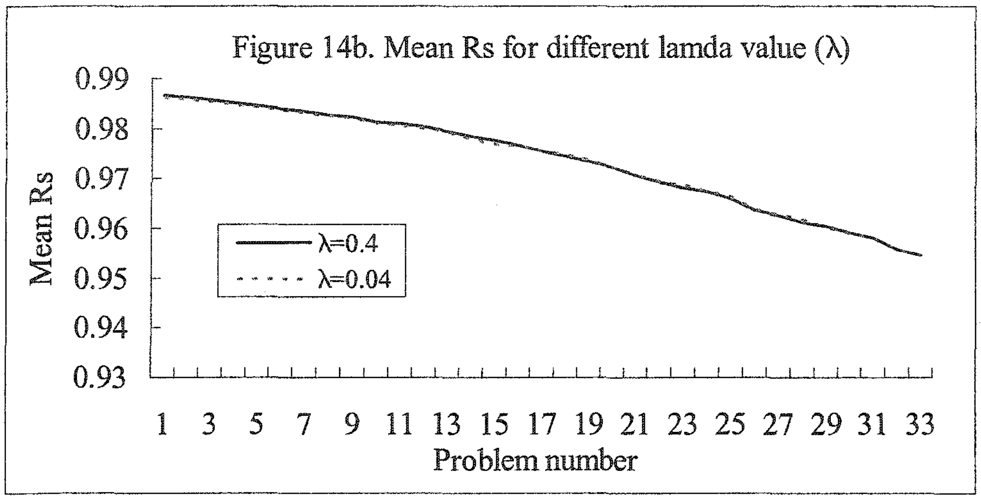
4.2.6 The influence of the penalty function

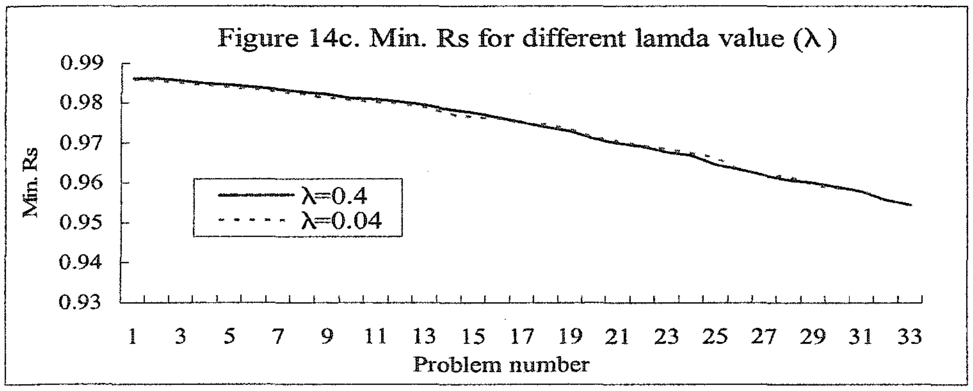
The penalty function used in ACSRAP algorithm is based on the dynamic "near-feasible threshold" (NFT) and severity parameters γ_1 and γ_2 for each constraint. Previous work has demonstrated substantial robustness to values of γ [46] and it was set to 2 for this research. According to chapter 3, The NFT is defined as

$$NFT = \frac{NFT_{\max}}{1 + \lambda \times NC}$$

NFT_{max} is the upper bond of cost or weight constraint among the feasible solutions in each iteration. *NC* is the total iteration and is set to 300. To investigate the influence of the NFT to the algorithm, two values of λ (0.04, 0.4) were tested and the results are shown in figure 14.







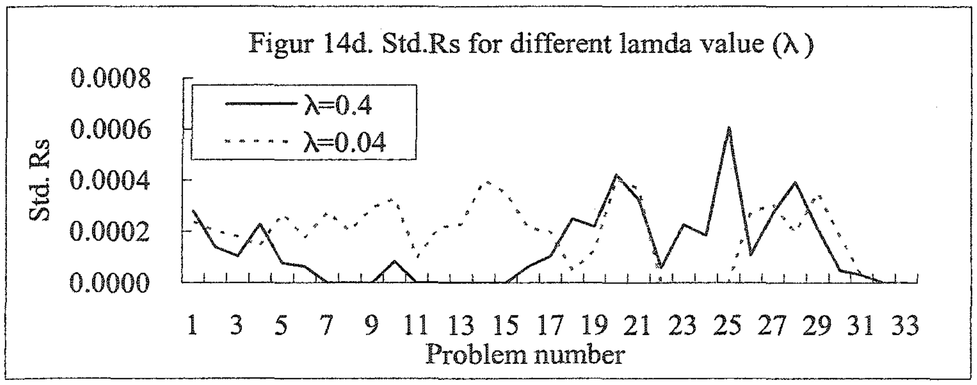


Figure 14 Influence of penalty function with different λ

It is observed that for highly constraint problems, $\lambda = 0.4$ outperforms $\lambda = 0.04$ in all measures; but for less constraint problems starting from problem number 17 where $NFT_{w\max} \leq 174$, the results with $\lambda = 0.04$ outperform those with $\lambda = 0.4$. Since the λ value influences the speed of the NFT approaching zero, the restrictions on selecting the λ is that the NFT does not approach zero either too quickly or too slowly.

In ACSRAP algorithm, an adaptive strategy for selecting the value of λ is applied with respect to the problem constraints: the default value of λ is set to 0.4. When $NFT_{w\max}$ or $NFT_{c\max}$ of the best feasible ant changes to a certain degree, the solution of problems is compared with that of $\lambda = 0.04$ during the first two runs and the λ with a better result was chosen.

4.2.7 The influence of local search strategies

As discussed in chapter 3, there are two types of local search in ACSRAP. The first local search happens for all ants in the colony during their solution construction phase. The other happens only for the best feasible ant in each iteration. Obviously, the first local search is more important and interesting because it is one of the most distinctive characteristics in ACS algorithms.

To investigate the effect of such strategy, three version of ACSRAP were invented and compared:

- ACSRAP-I: After ant selected the minimum components for each subsystem, the deterministic local search for best components with minimum cost and weight starts immediately.
- ACSRAP-I-Simple: only the first local search strategy is applied. There is no local search for the best feasible ant.

- ACSRAP-II, when redundancy in a subsystem exceeds a certain level (set to 3 in the algorithm), which is determined by used budget of system resource, ants select the best component with minimum resource consumption and replace the other components in the same subsystem one by one until this subsystem uses minimum resource budget to get better reliability. Both versions of ACSRAP-I and ACSRAP-II use second local search strategy.

The results of different local search strategies are shown in figure 15.

Figure 15a. Max.Rs of different local search strategies

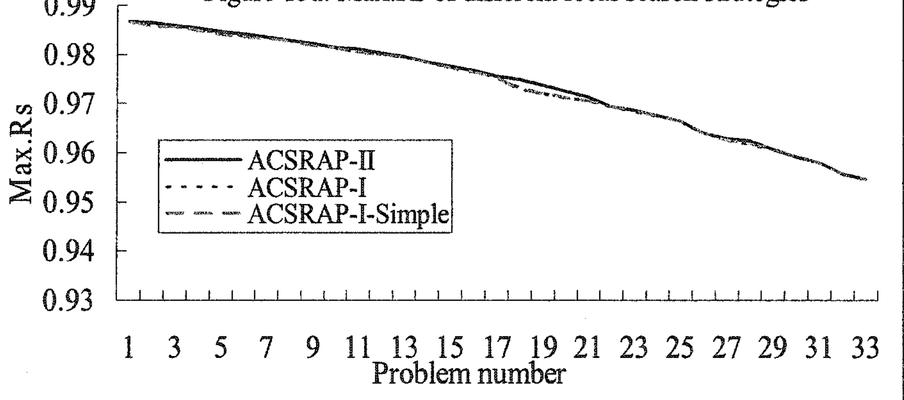
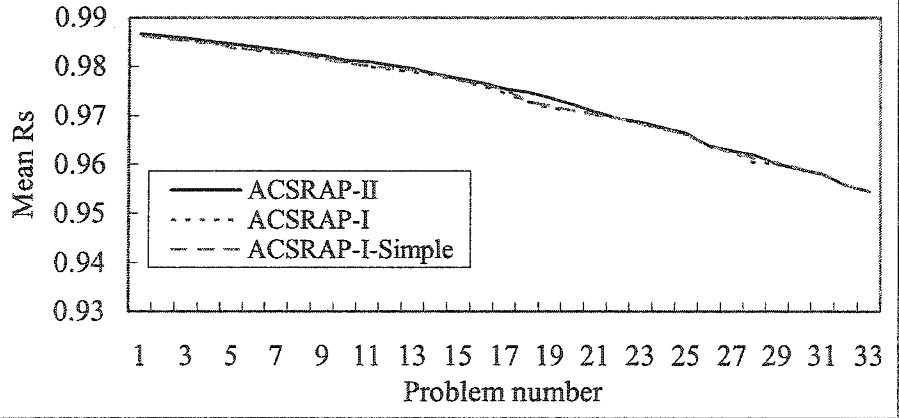
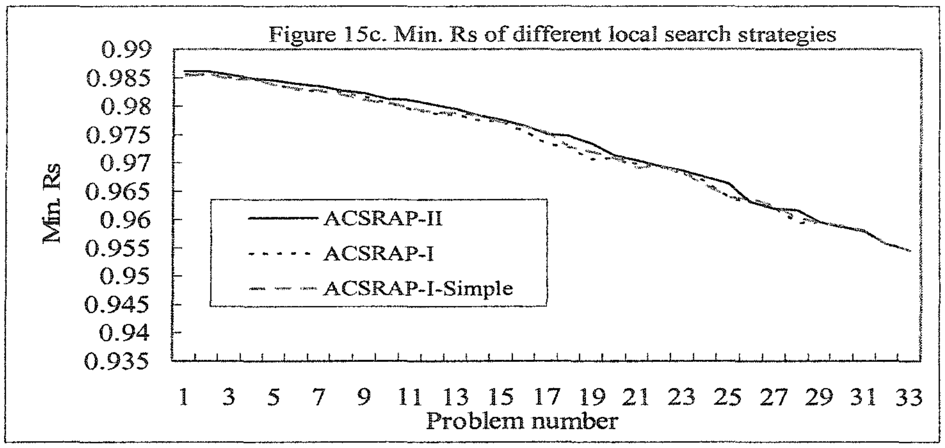


Figure 15b. Mean Rs of different local search strategies





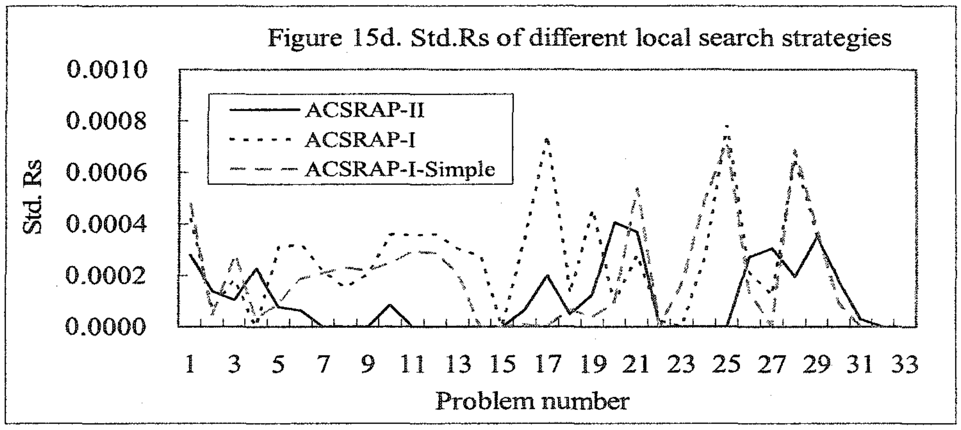


Figure 15 Influence of different local search strategies

It can be seen that the version of ACSRAP-II dominates the other versions in all performance measure, while the other two versions have similar results. So an appropriate local search method contributes the most on improving the solution quality. It leads the search to the right direction and avoids the stagnation of local optimum. Since the setting of the control level for each subsystem during local construction phase for ACSRAP-II has the best results, it is adopted as standard setting for the rest of this research.

4.3 Comparison of the multiobjective ACSRAP with GARAP and ACO-RAP [52]

In a multiobjective ACSRAP algorithm, the number of ants is set to (50, 100, 200). The results with 100 ants are better than those with 50, but do not differ from those with 200 ants. The dynamic memory of ranked feasible ants varied from 1 to 40, depending on the ant colony size. It was observed that a memory size of 20 with a 100-ant colony showed better results. Three different values of Q (1, 50, 100) were tested, but did not appear to have any influence on the algorithm. Q was then set to 1. q_0 is changed from 0.5 to 0.9 in favor of probabilistic exploration. Three different β values (0.3,0.5, 1) and three different ρ values (0.3, 0.6, 0.8) were tested as well as two different values of λ (0.04, 0.4). The default parameter values, (β, ρ, Q, λ) equal to (0.3, 0.6, 1, 0.4) with 20 ranked feasible ants produced better results. The algorithm was realized with Matlab version 6.5.0 on a Pentium IV 2.2GHz with 256RAM. 10 random runs with 300 iterations in each run were used for each of the 33-problem variations.

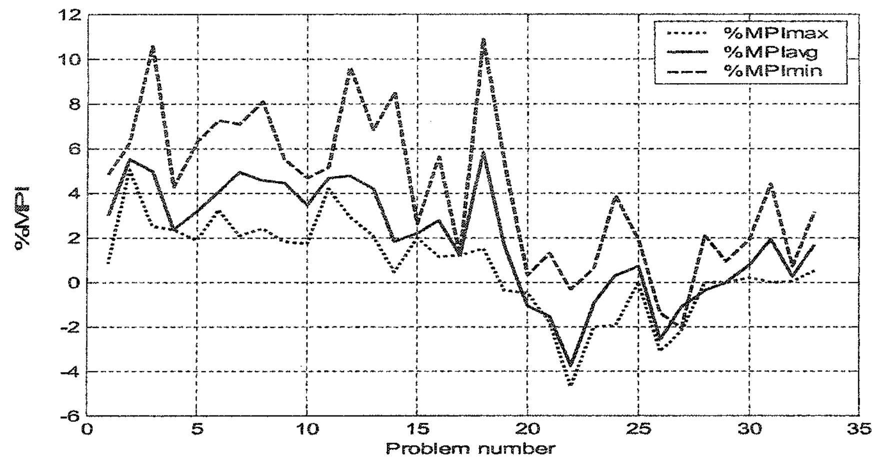
Table IV gives the results of ACSRAP and the results of GARAP [22] from Coit & Smith. The maximum, average and minimum Rs of the best solutions found among the 10 runs are presented as well as the standard deviation of the 10 final solutions. The %MPI values for maximum, average and minimum reliability over ten runs are compared and are shown in Figure 16. The standard deviation is an important measure of the robustness of the algorithms. Figure 17 shows a comparison of standard deviation of maximum system reliability between the GARAP and the multiobjective ACSRAP.

As the results in Table IV indicate, the multiobjective ACSRAP generally outperformed the GARAP. The ACSRAP won 20 of the 33 test problems in maximum system reliability (max. Rs), 25 of 33 test problem in average Rs, and 30 of 33 test problem in minimum Rs among 10 runs. In addition, it provided a lower standard deviation in all of the 33 test problems. ACSRAP thus generally yielded solutions with a higher reliability, and was more consistent. Since the multiobjective ACSRAP uses a constructive strategy instead of an improvement strategy as does the GARAP. It is not surprising that the ACSRAP is more consistent across the solutions.

Table IV

A comparison of the GARAP of Coit and Smith [22] and of the ACSRAP

No.	Cmax	Wmax	GA				ACS-multiobjective			
			1200 iterations				300 iterations			
			Max Rs	AvgRs	Min Rs	Std Dev	Max Rs	AvgRs	MinRs	Std Dev
1	130	191	0.9867	0.9862	0.9854	4.22E-04	0.9868	0.9866	0.9861	2.79E-04
2	130	190	0.9857	0.9855	0.9852	5.95E-04	0.9864	0.9863	0.9861	1.38E-04
3	130	189	0.9856	0.9850	0.9838	8.63E-04	0.9859	0.9858	0.9855	1.03E-04
4	130	188	0.9850	0.9848	0.9842	3.06E-04	0.9854	0.9852	0.9849	2.47E-04
5	130	187	0.9844	0.9841	0.9835	7.48E-04	0.9847	0.9846	0.9845	7.77E-05
6	130	186	0.9836	0.9833	0.9827	5.86E-04	0.9842	0.9840	0.9840	6.35E-05
7	130	185	0.9831	0.9826	0.9822	8.56E-04	0.9835	0.9835	0.9835	1.17E-16
8	130	184	0.9823	0.9819	0.9812	4.65E-04	0.9827	0.9827	0.9827	1.17E-16
9	130	183	0.9819	0.9814	0.9812	3.89E-04	0.9822	0.9822	0.9822	1.17E-16
10	130	182	0.9811	0.9806	0.9803	3.81E-04	0.9814	0.9813	0.9812	8.67E-05
11	130	181	0.9802	0.9801	0.9800	8.42E-04	0.9810	0.9810	0.9810	1.17E-16
12	130	180	0.9797	0.9793	0.9782	7.91E-04	0.9803	0.9803	0.9803	1.17E-16
13	130	179	0.9791	0.9786	0.9780	5.38E-04	0.9795	0.9795	0.9795	1.17E-16
14	130	178	0.9783	0.9780	0.9764	7.01E-04	0.9784	0.9784	0.9784	1.17E-16
15	130	177	0.9772	0.9771	0.9770	1.03E-03	0.9776	0.9776	0.9776	1.17E-16
16	130	176	0.9764	0.9760	0.9751	7.51E-04	0.9767	0.9767	0.9765	6.62E-05
17	130	175	0.9753	0.9753	0.9753	7.95E-04	0.9756	0.9756	0.9756	1.17E-16
18	130	174	0.9744	0.9732	0.9716	8.12E-04	0.9748	0.9748	0.9747	4.83E-05
19	130	173	0.9738	0.9732	0.9719	7.53E-04	0.9737	0.9736	0.9734	1.25E-04
20	130	172	0.9727	0.9725	0.9712	1.08E-03	0.9725	0.9722	0.9713	9.21E-05
21	130	171	0.9719	0.9712	0.9701	8.12E-04	0.9713	0.9708	0.9705	2.66E-05
22	130	170	0.9708	0.9705	0.9695	8.21E-04	0.9694	0.9694	0.9694	1.17E-16
23	130	169	0.9692	0.9689	0.9684	4.15E-04	0.9686	0.9686	0.9686	1.17E-16
24	130	168	0.9681	0.9674	0.9662	5.96E-04	0.9675	0.9675	0.9675	1.17E-16
25	130	167	0.9663	0.9661	0.9657	3.04E-04	0.9663	0.9663	0.9663	1.17E-16
26	130	166	0.9650	0.9647	0.9636	5.69E-04	0.9640	0.9638	0.9631	2.72E-04
27	130	165	0.9637	0.9632	0.9627	4.74E-04	0.9629	0.9628	0.9619	3.05E-04
28	130	164	0.9624	0.9620	0.9609	6.59E-04	0.9624	0.9619	0.9617	1.96E-04
29	130	163	0.9606	0.9602	0.9592	4.01E-04	0.9606	0.9602	0.9596	3.51E-04
30	130	162	0.9591	0.9587	0.9579	8.33E-04	0.9592	0.9590	0.9587	1.80E-04
31	130	161	0.9580	0.9572	0.9561	8.08E-04	0.9580	0.9580	0.9580	1.17E-16
32	130	160	0.9557	0.9556	0.9554	4.73E-04	0.9557	0.9557	0.9557	1.17E-16
33	130	159	0.9543	0.9538	0.9531	3.63E-04	0.9546	0.9546	0.9546	1.17E-16



%MPI = 100%*(ACSRAP - GARAP) / (1- GARAP)

Figure 16 A comparison of GARAP [22] and the ACSRAP

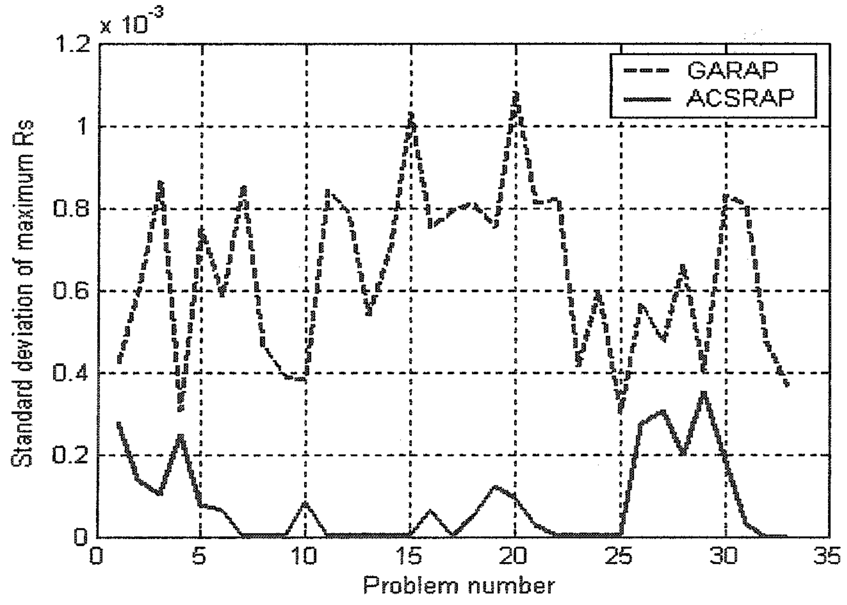
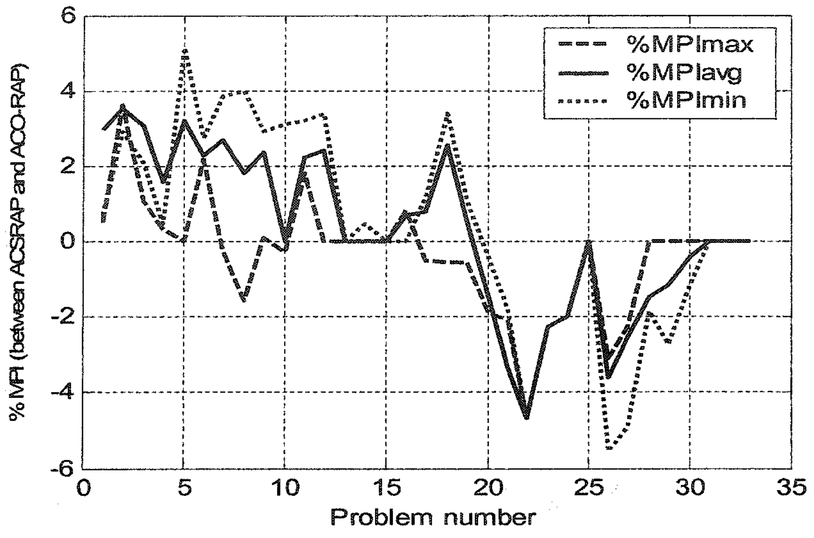


Figure 17 A comparison of standard deviation between GARAP [22] and ACSRAP

Table V compares the results between ACSRAP and ACO-RAP from Liang & Smith [35], and Figure 18 presents the %MPI values between ACSRAP and ACO-RAP for maximum, average and minimum system reliability. Although ACO-RAP won 3 cases in maximum

Rs, ACSRAP demonstrated better the average Rs and minimum Rs of 10 best solutions found among 10 runs. It can be said that the overall reliability performance of ACSRAP is similar to ACO-RAP.



%MPI = 100%*(ACSRAP - ACORAP)/(1-ACORAP)

Figure 18 A comparison of ACO-RAP [35] and the ACSRAP

An algorithm efficiency comparison is often made based on computer CPU time. But since the algorithms were run on were run on different computers with different operating systems and processors. It is not meaningful for this comparison. As an alternative, a comparison can be based on the number of iterations required, which gives the total evaluation of objective function. In many cases, this is more meaningful because it provides an absolute measure irrespective of how much faster the computer processing time becomes. For the GARAP, the stopping criterion was 1200 iterations with a population of 40. The ACO-RAP based on a maximum of 1000 iteration with ant colony size of 100, and The ACSRAP is based on a maximum 300 iterations with the same colony size as ACO-RAP. Thus, the multiobjective ACSRAP reduced the iteration time by approximately 37.5% of GARAP and by about 233% of ACO-RAP.

Table V

A comparison of the ACO-RAP of Liang &Smith [35] and of the ACSRAP

No.	Cs_{max}	Ws_{max}	ACO-RAP			ACS-multiobjective		
			max. 1000 iterations			max. 300 iterations		
			Max Rs*	AvgRs	Min Rs	Max Rs	AvgRs	MinRs
1	130	191	0.9867	0.9862	0.9860	0.9868	0.9866	0.9861
2	130	190	0.9859	0.9858	0.9857	0.9864	0.9863	0.9861
3	130	189	0.9858	0.9853	0.9852	0.9859	0.9858	0.9855
4	130	188	0.9853	0.9849	0.9848	0.9854	0.9851	0.9849
5	130	187	0.9847	0.9841	0.9837	0.9847	0.9846	0.9845
6	130	186	0.9838	0.9836	0.9835	0.9842	0.9840	0.9840
7	130	185	0.9835	0.9830	0.9828	0.9835	0.9835	0.9835
8	130	184	0.9830	0.9824	0.9820	0.9827	0.9827	0.9827
9	130	183	0.9822	0.9818	0.9817	0.9822	0.9822	0.9822
10	130	182	0.9815	0.9813	0.9806	0.9814	0.9813	0.9812
11	130	181	0.9807	0.9806	0.9804	0.9810	0.9810	0.9810
12	130	180	0.9803	0.9798	0.9796	0.9803	0.9803	0.9803
13	130	179	0.9795	0.9795	0.9795	0.9795	0.9795	0.9795
14	130	178	0.9784	0.9784	0.9783	0.9784	0.9784	0.9784
15	130	177	0.9776	0.9776	0.9776	0.9776	0.9776	0.9776
16	130	176	0.9765	0.9765	0.9765	0.9767	0.9767	0.9765
17	130	175	0.9757	0.9754	0.9753	0.9756	0.9756	0.9756
18	130	174	0.9749	0.9741	0.9738	0.9748	0.9748	0.9747
19	130	173	0.9738	0.9735	0.9731	0.9737	0.9736	0.9734
20	130	172	0.9730	0.9726	0.9714	0.9725	0.9722	0.9713
21	130	171	0.9719	0.9717	0.9710	0.9713	0.9708	0.9705
22	130	170	0.9708	0.9708	0.9708	0.9694	0.9694	0.9694
23	130	169	0.9693	0.9693	0.9693	0.9686	0.9686	0.9686
24	130	168	0.9681	0.9681	0.9681	0.9675	0.9675	0.9675
25	130	167	0.9663	0.9663	0.9663	0.9663	0.9663	0.9663
26	130	166	0.9650	0.9650	0.9650	0.9640	0.9638	0.9631
27	130	165	0.9637	0.9637	0.9637	0.9629	0.9628	0.9619
28	130	164	0.9624	0.9624	0.9624	0.9624	0.9619	0.9617
29	130	163	0.9606	0.9606	0.9606	0.9606	0.9602	0.9596
30	130	162	0.9592	0.9592	0.9592	0.9592	0.9590	0.9587
31	130	161	0.9580	0.9580	0.9580	0.9580	0.9580	0.9580
32	130	160	0.9557	0.9557	0.9557	0.9557	0.9557	0.9557
33	130	159	0.9546	0.9546	0.9546	0.9546	0.9546	0.9546

* Max Rs was calculated with the configuration of ACORAP [35], the Max Rs for case no.1 is 0.986745

The comparison of algorithm's capability to complex problem can be made based on the problems size. In GARAP, The problem size to RAP is larger than 7.6 x 10^{33}. The problem size handled by ACO-RAP is less than 6.45 x 10^{22}. The multiobjective ACSRAP easily solved RAP with search space larger than 4.3 x 10^{37}. Therefore, ACSRAP demonstrates superior performance than GARAP and ACO-RAP. Table VI shows the best system configuration for each problem.

4.4 Summary

ACS has previously been demonstrated to be a successful approach for many discrete optimization problems. However, its ability to provide sound solutions to the RAP under a multiobjective formulation had not yet been reported. In this research, we introduce the ACS algorithm based on a multiobjective formulation in order to solve the RAP. Through random search, constructive local move and long term dynamic memory strategy, the ACS efficiently builds a good solution for the RAP. When compared to the GARAP, the ACSRAP results in a better performance in terms of best solution found and reduced variation and great efficiency. When compared to the ACO-RAP, the ACSRAP results in a similar reliability performance but high efficiency and better capacity to handle large-scale problem. Meanwhile, ACSRAP algorithm demonstrated a better constructive strategy than ACO-RAP. It should be noted that the ACS algorithm reported here is rather simple, and that some features that are normally used effectively in complex problems, such as the candidate list or other local search techniques, are not incorporated in this paper. There are opportunities to improve on the effectiveness and the efficiency by considering the addition of these features to the ACS device here

Table VI

System configuration with maximum reliability

No.	Cs_{max}	Ws_{max}	Rs_{max}	Subsystem Configuration													
				1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	130	191	0.98681	333	11	111	2222	333	22	333	3333	12	112	11	4444	22	12
2	130	190	0.98642	333	11	111	2222	333	22	333	3333	22	112	11	4444	12	12
3	130	189	0.98592	333	11	111	2222	333	22	333	3333	13	112	13	4444	22	12
4	130	188	0.98538	333	11	111	2222	333	22	333	3333	13	122	13	4444	12	12
5	130	187	0.98469	333	11	111	2222	333	22	333	3333	23	122	13	4444	11	12
6	130	186	0.98415	333	11	111	2222	333	22	333	3333	13	112	13	4444	12	22
7	130	185	0.98346	333	11	111	2222	333	22	333	3333	23	112	13	4444	11	22
8	129	184	0.98272	333	11	111	222	333	22	333	3333	23	122	13	4444	11	22
9	130	183	0.98223	333	11	111	222	333	22	333	3333	33	122	13	4444	12	22
10	127	182	0.98142	333	11	111	222	333	22	333	3333	23	112	11	4444	11	22
11	129	181	0.98103	333	11	111	222	333	22	333	3333	33	112	11	4444	11	22
12	128	180	0.98029	333	11	111	222	333	22	333	3333	33	122	11	4444	11	22
13	126	179	0.97950	333	11	111	222	333	22	333	3333	33	122	13	4444	11	22
14	125	178	0.97840	333	11	111	222	333	22	333	3333	33	222	13	4444	11	22
15	126	177	0.97760	333	11	111	222	333	22	333	133	33	122	13	4444	11	22
16	124	176	0.97669	333	11	111	222	333	22	11	3333	33	122	13	4444	11	22
17	126	175	0.97559	333	11	111	222	333	22	11	3333	33	222	13	4444	11	22
18	124	174	0.97479	333	11	111	222	333	22	11	133	33	122	13	4444	11	22
19	123	173	0.97369	333	11	111	222	333	22	11	133	33	222	13	4444	11	22
20	121	172	0.97252	333	11	111	222	333	22	11	133	33	222	33	4444	11	22
21	120	171	0.97135	333	11	111	222	333	22	12	133	33	222	33	4444	11	22
22	120	170	0.96939	333	11	111	222	33	22	13	3333	33	222	13	4444	11	22
23	121	169	0.96859	333	11	111	222	33	22	13	133	33	122	13	4444	11	22
24	120	168	0.96750	333	11	111	222	33	22	13	133	33	222	13	4444	11	22
25	118	167	0.96634	333	11	111	222	33	22	13	133	33	222	33	4444	11	22
26	116	166	0.96395	333	11	111	222	33	22	13	133	33	222	33	144	11	22
27	117	165	0.96289	333	11	11	222	33	22	13	133	33	122	13	4444	11	22
28	115	164	0.96240	333	11	11	222	333	22	33	133	33	222	33	4444	11	22
29	114	163	0.96064	333	11	11	222	33	22	13	133	33	222	33	4444	11	22
30	115	162	0.95919	333	11	11	222	33	22	33	133	33	222	13	4444	11	22
31	113	161	0.95803	333	11	11	222	33	22	33	133	33	222	33	4444	11	22
32	112	160	0.95571	333	11	11	222	33	22	33	333	33	222	13	4444	11	22
33	110	159	0.95456	333	11	11	222	33	22	33	333	33	222	33	4444	11	22

CHAPTER 5

APPLICATION OF ACSRAP FOR THE RELIABILITY OPTIMIZATION OF MECHANICAL SYSTEM

To verify the application of the ACSRAP for the reliability optimization of mechanical system, this chapter begins with the reliability modeling for a reduction gear train system. The reliability optimization of gear train system under design constraints with ACSRAP algorithm is discussed and numerical results are compared with other approaches.

5.1 Modeling of gear train system

The optimization of a multi-speed gear train system problem introduces a number of challenges. Many high-performance power transmission applications (e.g. automotive and aerospace) require that the design of gear train system must satisfy:

- A compact system with minimal dimension.
- Smooth and quiet running with minimum noise.
- High reliability and long life of usage.
- Competitive cost.
- Easy fabrication and assembly.
- Compatibility with other elements of machines.

A gear pair is shown in figure 19. G1 represents the pinion gear on shaft Let h is the thickness of the gear; A is the center distance between two gears. T_w, T_p are the numbers of teeth on the wheel and pinion respectively, ω_p, ω_w are the speeds of the pinion (rpm) and wheel. The transmission ratio is given by :

$$ratio = \frac{T_w}{T_p} = \frac{\omega_p}{\omega_w} \tag{5.1}$$

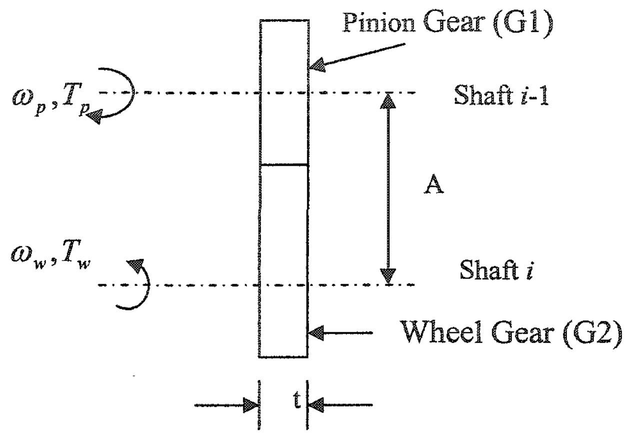


Figure 19 Schematic diagram of gear pair in stage *i* of gear train system

Before discussing the reliability optimization of gear train system, there are the following assumptions :

- Two modes, namely, good or fail, are considered for each gear pair.
- The gear train is idealized as a weakest –link kinematical chain, a concept analogous to the series system.
- The layout of the gears, the number of teeth on the different gears, the module, and the interconnection of various gear pairs of the gear train are known.
- The power transmitted by all the gear pairs is the same.
- The combination of the gear pairs in each stage is repeatable.
- All the random variables follow normal distribution.
- A design is considered to be safe and adequate if the probability of the failure of the gear train is less than or equal to a specified small quantity in each of the two failure modes.

A model of gear train system is shown in figure 20. There are two stages in this system, denoted as subsystem *i* (*i* =1, 2). The machine components, such as gear pairs, shaft including gear spline, gear key, bearing, etc. can be treated as the components of the system, and the interconnection of the components is shown in figure 21.

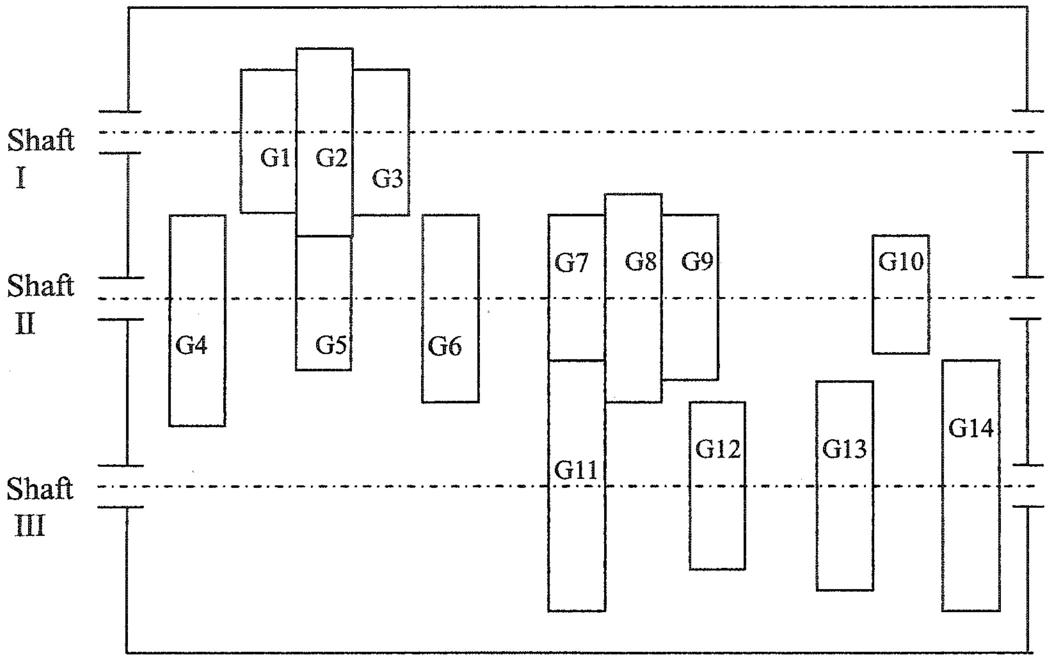


Figure 20 Modeling of gear train system

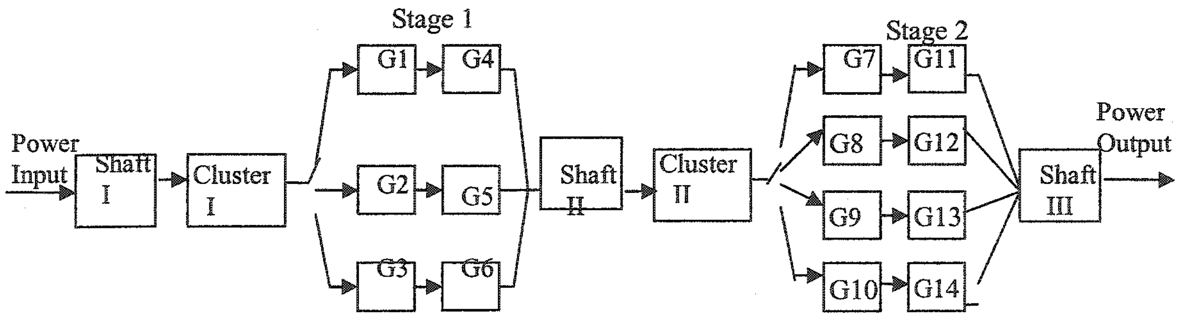


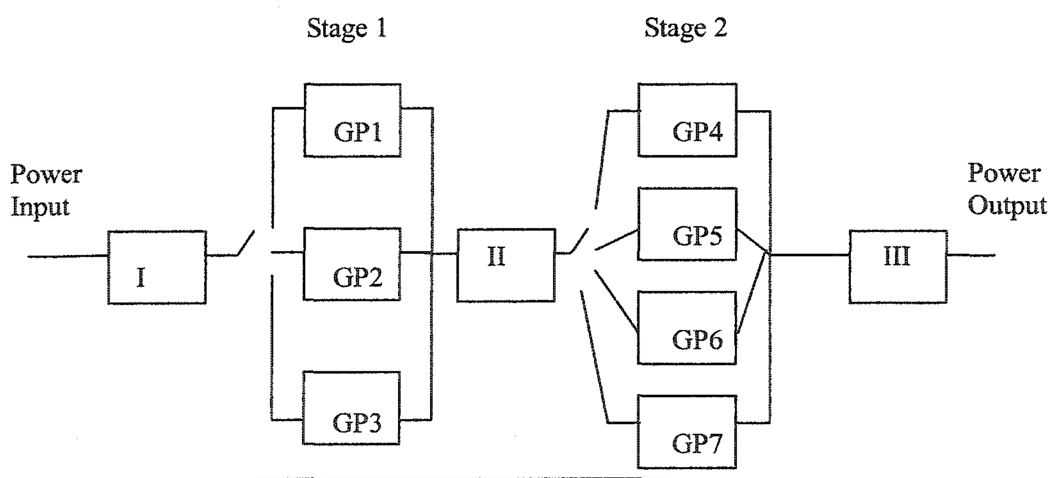
Figure 21 The connectivity of the components in gear train system

Let G1, G2 ... G14 represent the teeth number of each gear. For each stage, there are the following equations :

$$G1 + G4 = G2 + G5 = G3 + G6 \qquad \text{(for stage 1 between shaft I and II)} \qquad (5.2)$$

$$G7 + G11 = G8 + G12 = G9 + G13 = G10 + G14 \text{ (stage 2 between shaft II and III)(5.3)}$$

In this system, the failure is defined as zero power output from each stage of the system; therefore as long as there are one or more gear pairs in a stage, the system is still considered to be reliable to some degree (not a total system failure). Since one gear failure in a gear pair means the complete failure of gear pair, the gear pair can be defined as one component in the stage of gear train system. The equivalent schema bloc of this system can be represented in figure 22.



GP1:	Gear pair G1 - G4
GP2:	Gear pair G2 - G5
GP3:	Gear pair G3 - G6
GP4:	Gear pair G7 - G11
GP5:	Gear pair G8 - G12
GP6:	Gear pair G9 – G13
GP7:	Gear pair G10 – G14

Figure 22 The equivalent schema bloc for gear train system

Each gear pair from different stage can combine a speed output. In the system mentioned above, there are three gear pairs in stage 1 and four gear pairs in stage 2, and then the total combination number of speed output will be $\binom{3}{1} * \binom{4}{1} = 3 * 4 = 12$.

Suppose that each combination is independent from other combinations of the gear pairs. If one output combination is chosen, the other gear pairs in stages will become standby components (active) in the system. Therefore, we can calculate and improve the system reliability under cost and weight constraints for each output combination respectively with the same computational procedure.

5.2 Problem formulation

5.2.1 The expression of reliability, cost and weight of gear pair component

Consider the following case: a speed output combination of gear pair from different stage i (i =1, 2… s) of a gear train system is shown in figure 23. It is analogous to a series system with i stages or subsystems.

Let GP_{ij} represent a gear pair made from gears G_i^1 and G_i^2. The index j, $j \in (1, 2, \cdots m_i)$ represents different component choices for gear pair. So the reliability, cost and weight of gear pair GP_{ij} in each stage can be calculated as follows:

$$r_{ij} = r_{ij}^1 \cdot r_{ij}^2 \tag{5.4}$$

$$c_{ij} = c_{ij}^1 + c_{ij}^2 \tag{5.5}$$

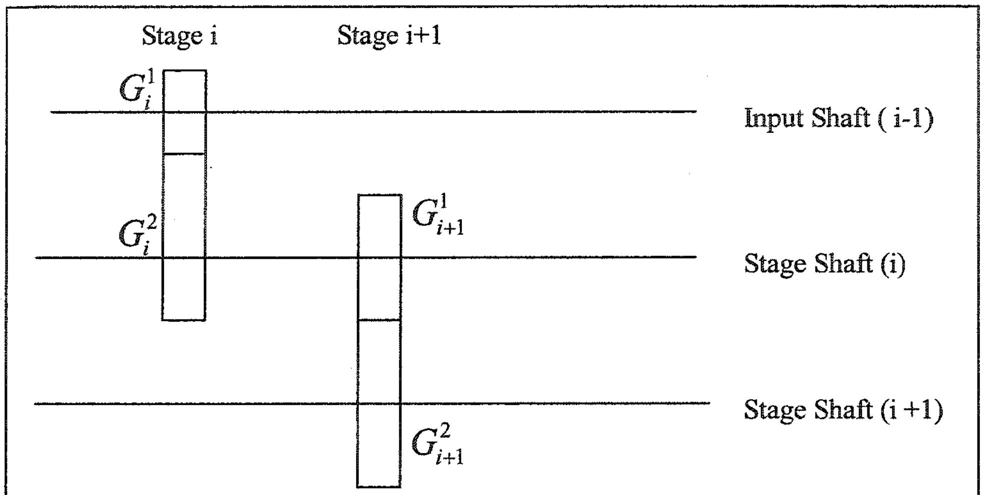
$$w_{ij} = w_{ij}^1 + w_{ij}^2 \tag{5.6}$$

$\forall i, i \in (1, 2, \cdots s)$ and $\forall j, j \in (1, 2, \cdots m_i)$

where r_{ij}^1, c_{ij}^1, w_{ij}^1 : reliability, cost, weight of gear G_i^1 in the stage i respectively.

r_{ij}^2, c_{ij}^2, w_{ij}^2 : reliability, cost, weight of gear G_i^2 in the stage i respectively.

r_{ij} , c_{ij}, w_{ij}: reliability, cost, weight of gear pair GP_{ij} in the stage i respectively



G_i^1 and G_i^2 represent gear 1 and 2 in gear pair of stage i

G_{i+1}^1 and G_{i+1}^2 represent gear 1 and 2 in gear pair of stage i+1

Figure 23 One combination of gear pairs of different stages

5.2.2 Gear pair configuration in transmission stage of gear train system

Suppose the gear pair in stage can be repeatable for improving the reliability of the stage, and there are m_i types of gear pair in stage i for choice. Then the combination of gear pairs in the gear train system becomes a series-parallel system with k-out-of-n: G subsystems whose configuration can be expressed by matrix form as described in chapter 3.

Let *PN* is the maximum redundancy of gear pairs can be put into the stage i, then the configuration of gear train system for one output combination can be expressed as a $PN \times i$ matrix. The element $x_{ki}\,(k = 1, 2, \cdots PN\,; i = 1, 2 \cdots, s)$ in the matrix represents a type of gear pair to be chosen, and $x_{ki} \in (1, 2, \cdots, m_i, m_{i+1})$ at the position k for stage i. An index m_{i+1} is assigned to the position where no component (defined as "blanks") to be used in the transmission stage i.

For example, consider a gear train system with 3 transmission stages, i.e., $s = 3$, and the types of gear pairs for an output combination in each stage are $m_1 = 4$, $m_2 = 3$, $m_3 = 4$ respectively, and the maximum redundancy level $PN = 5$. The following matrix of system configuration

$$\boldsymbol{x} = \begin{bmatrix} 3 & 1 & 4 \\ 3 & 2 & 4 \\ 3 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad (5.7)$$

represents a output combination of gear train system in which three of the third gear pairs used in parallel for the first transmission stage; one of the first gear pairs and two of the second gear pairs used in parallel for the second transmission stage and two of the fourth gear pairs used for the third transmission stage. Figure 24 represents a physical configuration of the gear pair combination in gear train system denoted by Equation (5.7).

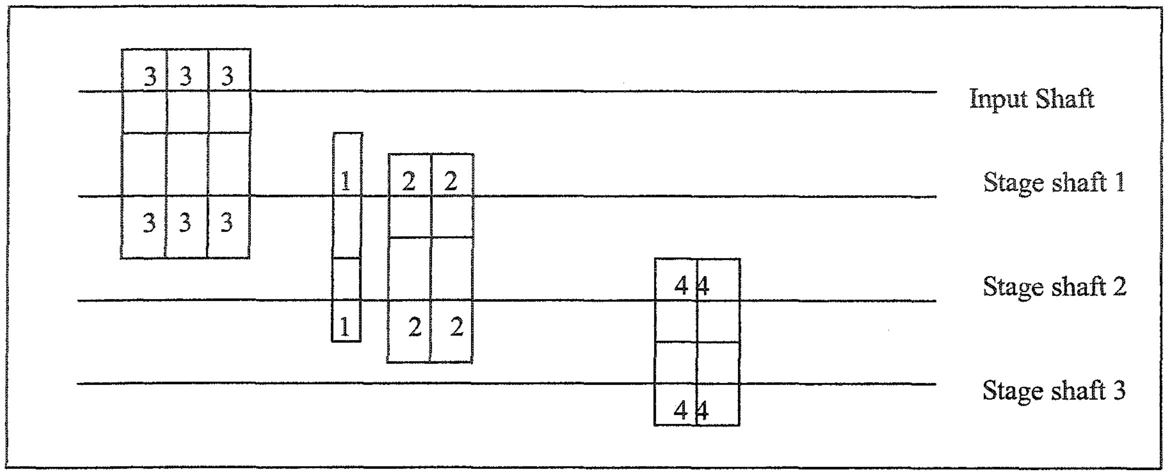


Figure 24 Configuration of the gear pair combination in gear train system

5.2.3 Formulation of gear train reliability optimization problem

The reliability optimization problem for gear train system can be formulated the same way as did in chapter 3. The total number n_i of redundant components in each stage i can be given by:

$$n_i = \sum_{k=1}^{PN} x_{ki} / x_{ki} \qquad x_{ki} \in (1,2,\dots m_i) \tag{5.8}$$

The reliability R_i, cost C_i and weight W_i of each stage can be represented by:

$$R_i(x|n_i) = 1 - \prod_{k=1}^{PN} (1 - r_{i\,x_{ki}}) \tag{5.9}$$

$$C_i(x|n_i) = \sum_{k=1}^{PN} c_{i\,x_{ki}} \tag{5.10}$$

$$W_i(x|n_i) = \sum_{k=1}^{PN} w_{i\,x_{ki}} \tag{5.11}$$

$$x_{ki} \in (1,2,\ldots m_i),\ k \in (1, 2,\cdots PN)\ and\ i \in (1, 2\cdots, s)$$

where $r_{i\,x_{ki}}$: reliability of the chosen type x_{ki} of gear pair in stage *i*.

$c_{i\,x_{ki}}$: cost of the chosen type x_{ki} of gear pair in stage *i*.

$w_{i\,x_{ki}}$: weight of the chosen type x_{ki} of gear pair in stage *i*.

n_i: total number of redundant gear pairs in subsystem *i*

Let $R_s(x)$, $C_s(x)$, $W_s(x)$ represent the reliability, cost and weight of overall system, the multiple objective reliability optimization of gear train system can be formulated as follows:

Maximize $\quad R_s(x) = \prod_{i=1}^{s} R_i(x)$

Minimize $\quad Cs(x) = \sum_{i=1}^{s} C_i(x)$ and $\quad Ws(x) = \sum_{i=1}^{s} W_i(x)$

Subject to $\quad Cs(x) = \sum_{i=1}^{s} C_i(x) = \sum_{i=1}^{s} \sum_{k=1}^{PN} c_{i\,x_{ki}} \leq Cs_{\max}$,

$$Ws(x) = \sum_{i=1}^{s} W_i(x) = \sum_{i=1}^{s} \sum_{k=1}^{PN} w_{i\,x_{ki}} \leq Ws_{\max}$$

$$R_s(x) = \prod_{i=1}^{s} [1 - \prod_{k=1}^{PN} (1 - r_{i x_{ki}})] \geq Rs_{\min}$$

$$p_i \leq n_i \leq PN$$

5.3 Past studies and their shortcomings

Past efforts to solve optimal reliability design problem of mechanical system and gear train were formulated as a multi-objective problem [49] [50], and solved by "a priori" methods such as goal programming and fuzzy logic method to get Pareto optimal solution. The formulation of the multiple objective optimizations is given as follows:

Minimize $[-f_1, f_2, f_3]$

With $f_1 = R_s(x) = \prod_{i=1}^{s} R_i(x)$

$$f_2 = Cs(x) = \sum_{i=1}^{s} C_i(x)$$

$$f_3 = Cs(x) = \sum_{i=1}^{s} C_i(x)$$

Subject to $Cs(x) = \sum_{i=1}^{s} C_i(x) = \sum_{i=1}^{s} \sum_{k=1}^{PN} c_{i x_{ki}} \leq Cs_{\max}$

$$Ws(x) = \sum_{i=1}^{s} W_i(x) = \sum_{i=1}^{s} \sum_{k=1}^{PN} w_{i x_{ki}} \leq Ws_{\max}$$

$$R_s(x) = \prod_{i=1}^{s} R_i(x) \geq Rs_{\min}$$

$$p_i \leq n_i \leq PN$$

$$x_{ki} \in (1,2,\ldots m_i),\ k \in (1, 2, \cdots PN)\ and\ i \in (1, 2 \cdots, s)$$

To apply the classic methods such as goal programming, or fuzzy logic, at least the following procedures must be established:

- Convert all conflict objectives into the same type of objective function (either all are of minimization type or of maximization type). One common way to convert a maximization problem into a minimization problem is to use the inverse function.
- A weight vector or a preference vector in order of importance level of objectives, must be required before formulate the problem. Utility functions and goal programming are most popular approaches in such "a priori" methods.
- When the objectives and design constraints are not known precisely, membership functions must be selected to characterize and quantify the fuzzy goals for objective functions and the fuzzy design constraints.

Although such procedures enable the use of classic optimization techniques to be applied, the final obtained solutions may not necessarily correspond to the true optimal solution of the overall problem. It is important to note that there are at least the following difficulties with the above the approaches:

- Although the optimization of a converted objective should result in the same optimal solution as that would be obtained by optimizing the original objective in the case of a single objective problem; the same may not be true for multi-objective optimization. Such conversions do not emphasize the complete range of the transformed objective uniformly. Thus, a number of well-distributed or trade-off solutions in the original objective space may be difficult to obtain with a uniformly set of weight vectors used in the converted objective space by classic generating approach [50], such as such as a repetitive application of the weighted-sum approach [51].
- Since such a weight or preference vector scalarizes multiple objectives into a

single objective, the outcome of the optimization process is usually a single optimal solution. In order to obtain a set of so-called Pareto-optimal solutions for multi-objective optimization, these methods must have to be applied many times with different weight or preference vectors. However, for a moderate size problem, these methods are time consuming and needs a lot of computational capacity.

- For a complex problem with multiple objectives, it is not easy to determine the correct utility function for the scaling procedure. Similarly, for the fuzzy logic optimization, decision maker still need to rank the objectives with linguistic hedge. And it is also not easy to establish the correct membership functions for fuzzy objectives & constraints.

5.4 ACSRAP for reliability optimization problem of gear train system

The solution generation procedure and methodology of ACSRAP described in chapter 3 can be used for the multi-objective reliability optimization of gear train system. The procedures for the multi-objective reliability optimization of gear train system are as follows:

i. By using single objective formulation as described in chapter 3, ACSRAP can obtain the upper and lower boundary of system reliability ($Rs_{\min}$ *and* $Rs_{\max}$), system cost ($Cs_{\min}$ *and* $Cs_{\max}$) and system weight ($Ws_{\min}$ *and* $Ws_{\max}$) respectively.

ii. With given system cost and weight constraints, multi-objective ACSRAP generates sets of non-dominated solutions on the constraint boundary by use of adaptive search, and then chooses the best optimal feasible solution based on the preference of system reliability. This solution becomes a candidate solution with maximum system reliability subject to the usable recourse level. The achievement factors (f_{Rs}, f_{Cs}, f_{Ws}) are used to represent how much the solutions realize the goals.

$$f_{Rs}(\%) = 100 \times \frac{Rs - Rs_{\max}}{Rs_{\min} - Rs_{\max}}$$

$$f_{Cs}(\%) = 100 \times \frac{Cs_{\max} - Cs}{Cs_{\max} - Cs_{\min}} \tag{5.13}$$

$$f_{Ws}(\%) = 100 \times \frac{Ws_{\max} - Ws}{Ws_{\max} - Ws_{\min}}$$

iii. When the decision makers do not satisfy the obtained achievements to the goals. they can reduce weight or cost constraints at system level with respect to trade-off or preference information and run ACSRAP program again until they get overall satisfactory solution.

iv. If the given system weight or cost constraints are not sufficient to construct a feasible solution, i.e. unable to get minimum reliability requirement or to get minimum components for functional requirement of system, ACSRAP will remind the decision maker to increase the system cost or weight.

From the procedures mentioned above, we can find out that multi-objective ACSRAP is a non-dominated interactive approach for multi-objective optimization. There is no need for "a priori" preference information; there are less restrictive assumptions as compared to methods described previously. With these strategies, some of the computational difficulties associated with the classical approaches can be alleviated using ACS optimization approach, and a more systematic, flexible, and a combined optimization task can be achieved.

5.5 Test problems

The reliability optimization for a gear train system was simulated using Fuzzy Logic method by Quy Nguyen [49]. He specified a gear train system with four transmission stages. The minimum components no.(p_i) is 2, and the maximum component no. (PN) is 5 in each stage. There are several component types (gear pair) for choice in each stage. Component types in each stage can be identical or different. The input data of reliability, cost and weight for gear pairs in each stage are listed in table VII. The system cost constraint Cs_{max} varies from 60 to 75 and the weight constraint Ws_{max} varies from 120 to 150. There are total 19 variation instances of the problem with different system constraints (cost and weight) and system reliability requirement which are listed in Table VIII.

Table VII

Input data of the gear pairs in different stages

Gear Pair (j)	Gears (GP_{ij})	Stage 1			Stage 2			Stage 3			Stage 4		
		r_1	c_1	w_1	r_2	c_2	w_2	r_3	c_3	w_3	r_4	c_4	w_4
1	G1	0.90	1	3	0.94	3	5	0.92	4	8	0.87	1	6
	G2	0.95	2	8	0.79	2	4	0.90	5	7	0.85	5	4
	GP_{i1}	0.855	3	11	0.743	5	9	0.828	9	15	0.74	6	10
2	G1	0.85	2	7	0.98	2	5	0.99	3	9	0.95	3	5
	G2	0.83	3	5	0.9	4	6	0.85	4	5	0.97	2	5
	GP_{i2}	0.706	5	12	0.882	6	11	0.842	7	14	0.922	5	10
3	G1	0.94	2	4	0.93	1	4	0.95	4	6	0.94	5	9
	G2	0.99	3	5	0.94	1	10	0.82	3	5	0.91	6	6
	GP_{i3}	0.931	5	9	0.874	2	14	0.779	7	11	0.855	11	15
4	G1	0.91	4	7	0.9	3	5	0.99	3	5	0.96	5	7
	G2	0.81	3	4	0.87	4	6	0.92	4	7	0.9	4	6
	GP_{i4}	0.737	7	11	0.783	7	11	0.911	7	12	0.864	9	13
5	G1	0.97	2	8	0.93	2	3	0.91	2	2	0.96	5	6
	G2	0.83	4	6	0.98	3	4	0.93	1	9	0.85	4	6
	GP_{i5}	0.805	6	14	0.9114	5	7	0.846	3	11	0.816	9	12

Table VIII

System constraints and reliability requirement of gear train system

Problem no.	System cost limit (Cs_{max})	System weight limit (Ws_{max})	System reliability requirement (Rs_{min})
1	40	115	0.85
2	55	125	0.90
3	65	130	0.95
4	60	120	0.98
5	60	130	0.98
6	60	140	0.98
7	60	150	0.98
8	65	120	0.98
9	65	130	0.98
10	65	140	0.98
11	65	150	0.98
12	70	120	0.98
13	70	130	0.98
14	70	140	0.98
15	70	150	0.98
16	75	120	0.98
17	75	130	0.98
18	75	140	0.98
19	75	150	0.98

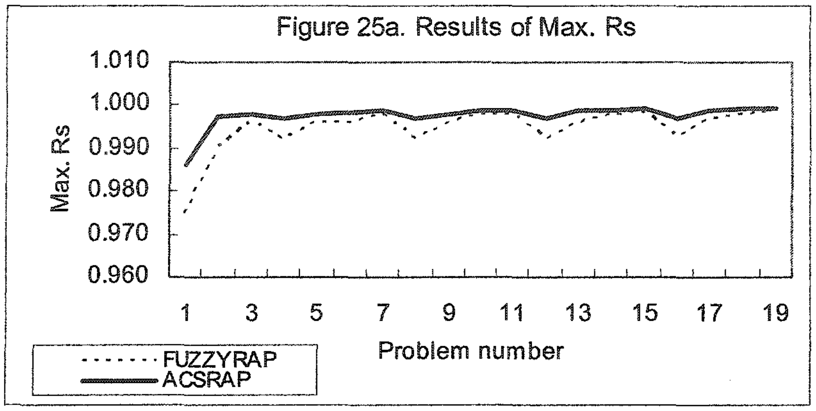
5.6 Numerical results and analysis

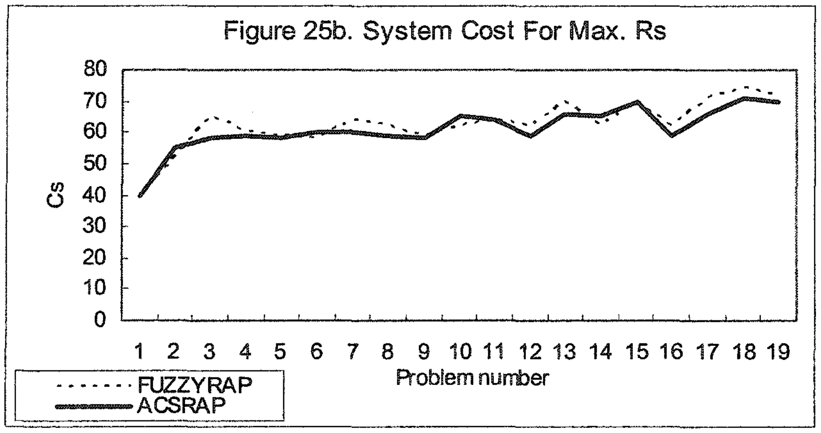
Four types of objectives are simulated with ACSRAP algorithm, namely:

- Maximize system reliability subject to limited system cost and weight.
- Minimize system weight subject to limited system cost and the requirement of system reliability.
- Minimize system cost subject to limited system weight and the requirement of system reliability.
- Obtain Perato optimal solution for multi-objective reliability optimization subject to limited system cost and weight and reliability requirement.

5.6.1 Maximizing system reliability

Figure 25 show the simulation results for the objective of maximizing the system reliability given cost and weight constraints at system level. Figure 25a demonstrates the maximum reliability obtained with cost and weight constraints at system level. Figure 25b and Figure 25c show the system cost and weight used to obtain the maximum *Rs*. From these results, it can be found out that the results of maximizing system reliability $Rs_{\max}$ using ACSRAP outperform those of FUZZYRAP for all the 19 variations of the problem with similar system cost and weight. This means that ACSRAP can fulfill the goal with better consistency than FUZZYRAP. The system configuration with maximum reliability is listed in Table IX.





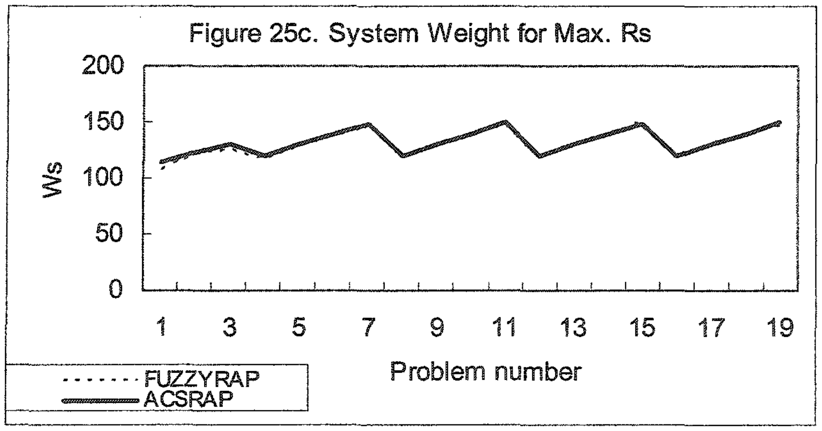


Figure 25 Maximum Rs between FUZZYRAP and ACSRAP

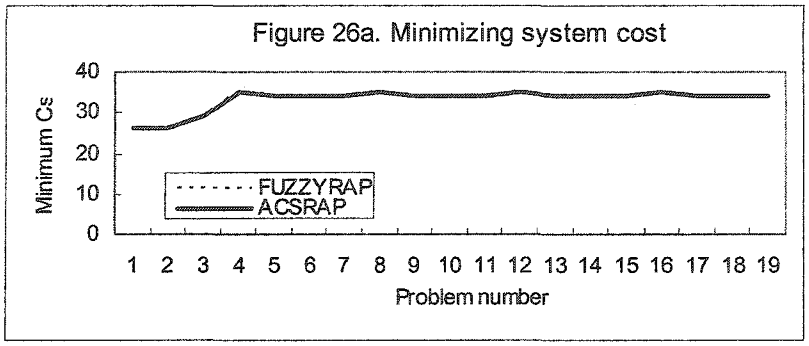
Table IX

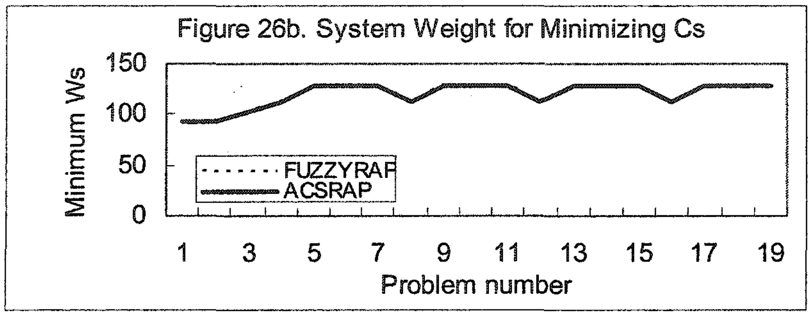
Configuration of gear train system with maximum reliability

No.	FUZZYRAP			ACSRAP						
	Cs	Ws	Max.Rs	Cs	Ws	Max.Rs	Configuration			
							1	2	3	4
1	40	107	0.9749	40	114	0.9861	111	355	555	22
2	52	121	0.9899	55	124	0.9973	133	555	5555	222
3	65	125	0.9963	58	130	0.9977	333	355	4555	222
4	60	116	0.9919	59	120	0.9968	333	355	445	222
5	59	129	0.9959	58	130	0.9977	333	355	4555	222
6	58	138	0.9961	60	140	0.9985	333	2355	5555	222
7	64	147	0.9981	60	149	0.9987	1133	3555	5555	222
8	62	118	0.9924	59	120	0.9968	333	355	445	222
9	59	129	0.9959	58	130	0.9977	333	355	4555	222
10	62	140	0.9977	65	140	0.9988	1333	5555	5555	222
11	64	147	0.9981	64	150	0.9990	1133	3555	4555	222
12	62	118	0.9924	59	120	0.9968	333	355	445	222
13	70	130	0.9961	66	130	0.9988	333	5555	4555	222
14	62	140	0.9977	65	140	0.9990	333	5555	55555	222
15	70	150	0.9982	70	149	0.9992	1333	3555	4455	222
16	62	118	0.9928	59	120	0.9968	333	355	445	222
17	71	130	0.9968	66	130	0.9988	333	5555	4555	222
18	74	139	0.9979	71	140	0.9992	333	5555	4555	2222
19	72	147	0.9986	70	150	0.9995	333	5555	55555	2222

5.6.2 Minimizing system cost

The minimum system cost can be found by running ACSRAP with gradually reduced cost constraints. Figure 26 and Table X demonstrates the simulation results for the objective of minimizing the system cost subject to limited system weight and the requirement of system reliability. The ACSRAP obtains the same results of minimum system cost as FUZZYRAP.





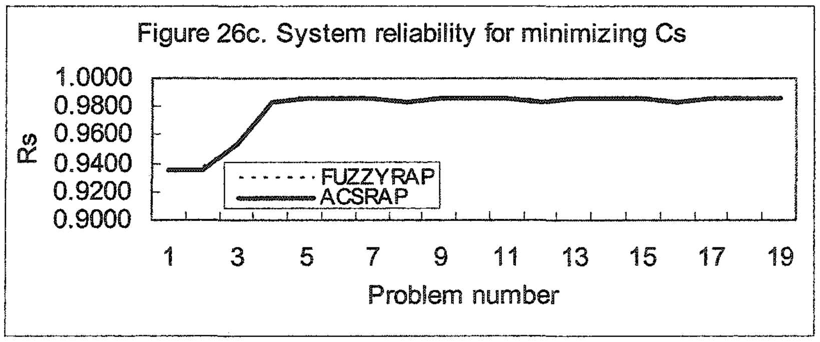


Figure 26 Minimum Cs between FUZZYRAP and ACSRAP

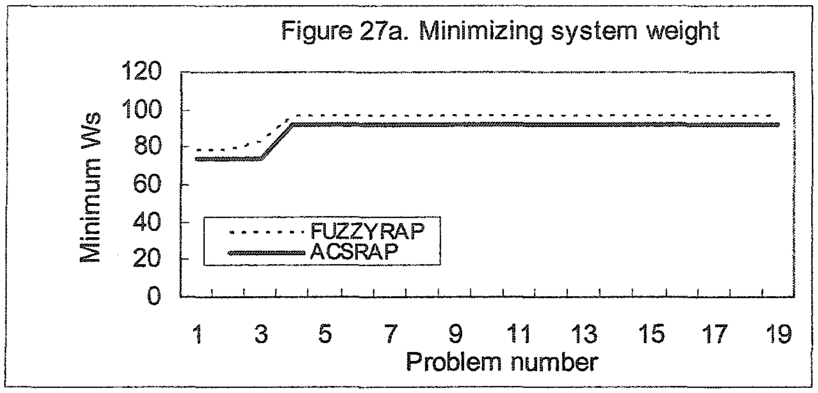
Table X

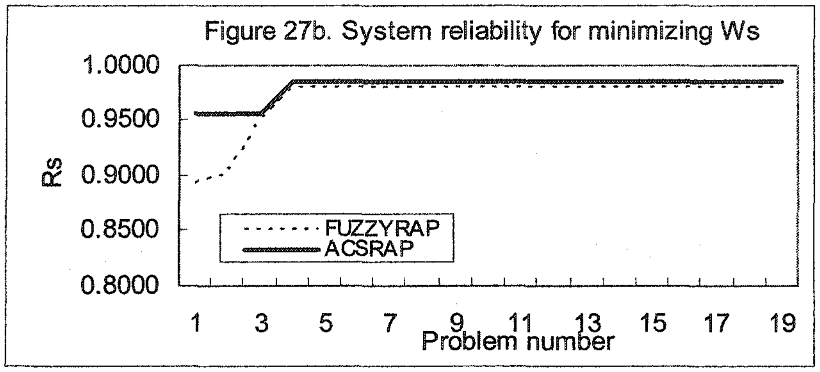
Configuration of gear train system with minimum cost

No.	FUZZY-RAP			ACS-RAP						
	Cs	Ws	Max.Rs	Cs	Ws	Max.Rs	Configuration			
							1	2	3	4
1	26	92	0.9349	26	92	0.9349	11	33	55	22
2	26	92	0.9349	26	92	0.9349	11	33	55	22
3	29	103	0.9541	29	103	0.9541	11	33	555	22
4	35	113	0.9836	35	113	0.9836	33	333	555	22
5	34	128	0.9853	34	128	0.9853	111	333	555	22
6	34	128	0.9853	34	128	0.9853	111	333	555	22
7	34	128	0.9853	34	128	0.9853	111	333	555	22
8	35	113	0.9836	35	113	0.9836	33	333	555	22
9	34	128	0.9853	34	128	0.9853	111	333	555	22
10	34	128	0.9853	34	128	0.9853	111	333	555	22
11	34	128	0.9853	34	128	0.9853	111	333	555	22
12	35	113	0.9836	35	113	0.9836	33	333	555	22
13	34	128	0.9853	34	128	0.9853	111	333	555	22
14	34	128	0.9853	34	128	0.9853	111	333	555	22
15	34	128	0.9853	34	128	0.9853	111	333	555	22
16	35	113	0.9836	35	113	0.9836	33	333	555	22
17	34	128	0.9853	34	128	0.9853	111	333	555	22
18	34	128	0.9853	34	128	0.9853	111	333	555	22
19	34	128	0.9853	34	128	0.9853	111	333	555	22

5.6.3 Minimizing system weight

The minimum system weight can be obtained by running the ACSRAP with gradually reduced weight constraint at system level. The simulation results for the objective of minimizing the system weight subject to limited system cost and the requirement of system reliability are shown in figure 27 and Table XI. Here again, the ACSRAP outperforms the FUZZYRAP with less system weight and better system reliability, and ACSRAP shows better consistency for the solution than FUZZYRAP.





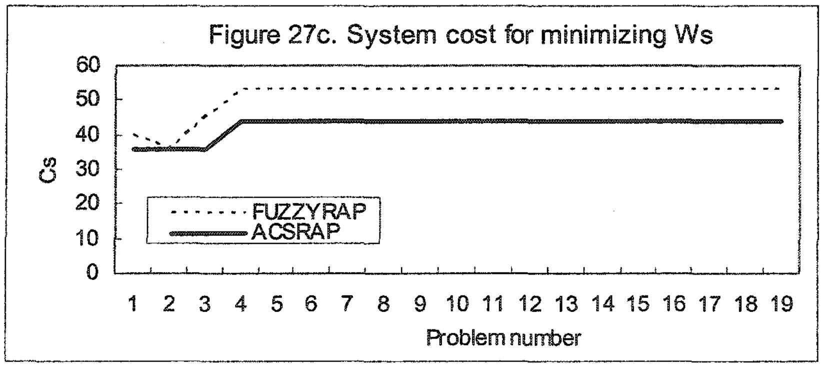


Figure 27 Minimum Ws between FUZZYRAP and ACSRAP

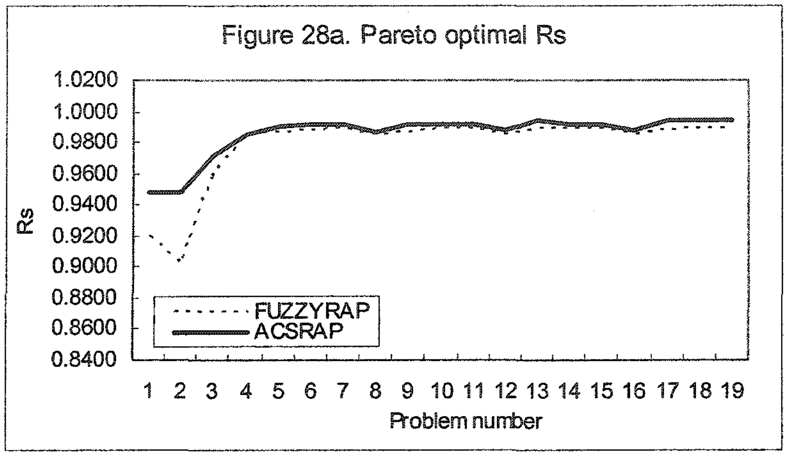
Table XI

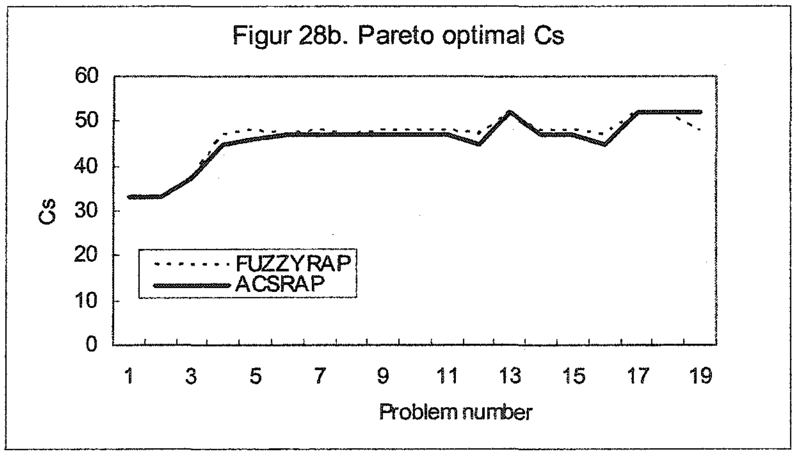
Configuration of gear train system with minimum weight

No.	FUZZY-RAP			ACS-RAP						
	Cs	Ws	Rs_{max}	Cs	Ws	Rs_{max}	Configuration			
							1	2	3	4
1	40	78	0.8924	36	74	0.9565	33	55	55	22
2	36	78	0.9019	36	74	0.9565	33	55	55	22
3	45	82	0.9516	36	74	0.9565	33	55	55	22
4	53	96	0.9800	44	92	0.9846	33	555	555	22
5	53	96	0.9800	44	92	0.9846	33	555	555	22
6	53	96	0.9800	44	92	0.9846	33	555	555	22
7	53	96	0.9800	44	92	0.9846	33	555	555	22
8	53	96	0.9800	44	92	0.9846	33	555	555	22
9	53	96	0.9800	44	92	0.9846	33	555	555	22
10	53	96	0.9800	44	92	0.9846	33	555	555	22
11	53	96	0.9800	44	92	0.9846	33	555	555	22
12	53	96	0.9800	44	92	0.9846	33	555	555	22
13	53	96	0.9800	44	92	0.9846	33	555	555	22
14	53	96	0.9800	44	92	0.9846	33	555	555	22
15	53	96	0.9800	44	92	0.9846	33	555	555	22
16	53	96	0.9800	44	92	0.9846	33	555	555	22
17	53	96	0.9800	44	92	0.9846	33	555	555	22
18	53	96	0.9800	44	92	0.9846	33	555	555	22
19	53	96	0.9800	44	92	0.9846	33	555	555	22

5.6.4 Pareto optimal solution for multi-objective reliability optimization

By reducing the cost and weight constraints at system level according to the trade-off preference of the decision maker, ACSRAP can get Pareto optimal solution for the reliability problem of gear train system. Figure 28 shows the simulation results of Perato optimal solution obtained by ACSRAP and FUZZYRAP respectively for gear train system. It can be found out that ACSRAP obtain better system reliability *Rs* with lower system cost *Cs* and system weight *Ws* than FUZZYRAP for 19 variations of the problem.





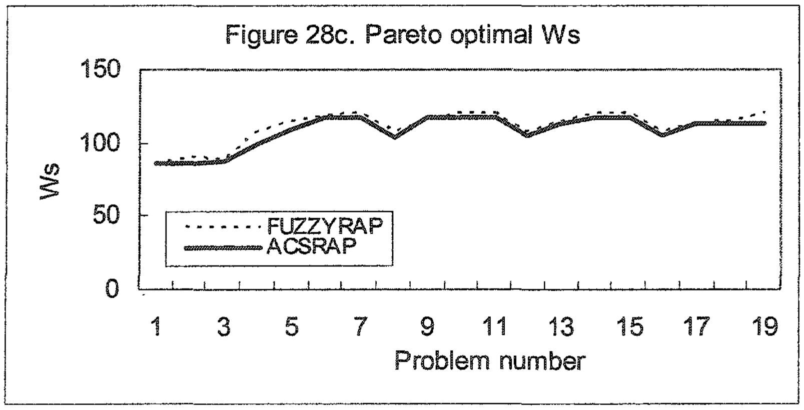


Figure 28 Pareto optimal solutions between ACSRAP and FUZZYRAP

The system configurations for Pareto solution are listed in Table XII.

Table XII

Configuration of gear train system for Pareto optimal solution

No.	FUZZY-RAP			ACS-RAP						
	Cs	Ws	Rs_{max}	Cs	Ws	Rs_{max}	Configuration			
							1	2	3	4
1	33	86	0.9199	33	86	0.9478	11	35	45	22
2	33	90	0.9021	33	86	0.9478	11	35	45	22
3	37	89	0.9585	37	87	0.9710	13	55	555	22
4	47	108	0.9854	45	100	0.9859	33	355	455	22
5	48	115	0.9868	46	109	0.9899	33	355	555	222
6	47	118	0.9874	47	117	0.9911	33	335	455	222
7	48	120	0.9896	47	117	0.9911	33	335	455	222
8	47	108	0.9854	47	103	0.9867	133	555	555	22
9	48	115	0.9862	47	117	0.9911	33	335	455	222
10	48	120	0.9896	47	117	0.9911	33	335	455	222
11	48	120	0.9896	47	117	0.9911	33	335	455	222
12	47	108	0.9854	45	105	0.9879	113	555	555	22
13	52	114	0.9895	52	113	0.9943	133	555	555	222
14	48	120	0.9896	47	117	0.9911	33	335	455	222
15	48	120	0.9896	47	117	0.9911	33	335	455	222
16	47	108	0.9854	45	105	0.9879	113	555	555	22
17	52	113	0.9883	52	113	0.9943	133	555	555	222
18	52	114	0.9895	52	113	0.9943	133	555	555	222
19	48	120	0.9896	52	113	0.9943	133	555	555	222

5.6.5 CPU time of simulation

For the above mentioned gear train system, with given system cost and weight, ACSRAP uses only 20 iterations and cost about 15 seconds to get reliability result on a PIII 500MHz computer. For each problem, it only takes several run of ACSRAP to get results. Compared with FUZZYRAP which used 30,000 – 100,000 iterations to get result, ACSRAP shows a better efficiency.

5.7 Summary

This chapter demonstrated the versatility and effectiveness of ACSRAP for the multiple objective reliability optimization problems of mechanical system. Compared with classic methods, ACSRAP uses interactive approach and obtains the same or better solutions without any major change in its parameter setting. The advantage of ACSRAP algorithm for such interactive approach include that there is no need for “a priori” preference information, there are less restrictive assumptions as compared to methods described previously, and solutions obtained have a better prospect of being implemented. The disadvantages may consist of no guarantee that the preferred solution can be obtained within a finite number of interactive iterations and more effort is required from the decision maker.

CONCLUSIONS

ACS has previously been demonstrated to be a successful approach for many discrete optimization problems. However, its ability to provide sound solutions to the reliability and redundancy optimization problems had not yet been reported thoroughly. This research introduced ACSRAP algorithm under multiobjective formulation to solve RAP for general k-out-of-n G system and to solve the reliability optimization problems of mechanical system. The results has proven that multiobjective Ant Colony System (ACS) algorithm is robust and versatile in solving well know NP-Hard combinatorial problems.

When using meta-heuristic methods, the choice of appropriate parameters always plays an important role in obtaining satisfactory solution quality. A detailed characteristic study of parameter setting of ACS for RAP problem is presented in this research.

Generally, the entire procedure of a multiobjective ACS algorithm can be summarized as follows: Starting with a good representation of the problem, it is then followed by the choice of the state transition rule which tries to balance exploration of probabilistic search and exploitation of local heuristic information. Heuristic information is used to help ants find good component choice. During the solution construction process, which is the most important phase in multiobjective ACS algorithm, local search technique is integrated into the local constructive strategy under multiobjective formulation to build the subsystem configuration. These random search plus deterministic locals move strategies help ants to build solution effectively and very quickly. After all ants construct solutions, local search is employed, which helps explore the local optimum or near local optimum areas. A dynamic penalty function is employed to integrated local search results for pheromone updating. Finally, local updating rule and global updating rule of pheromone on trails with dynamic long-term memory are used so as to guide future colonies toward the right direction of the search space.

Through random search, constructive local move and long term dynamic memory strategy, ACS efficiently and effectively built good solution for reliability optimization problems. When compared to GARAP or FUZZYRAP, ACSRAP results in a better performance in terms of best solution found and reduced variation and great efficiency.

For future research, the multi-colony ACS approach with parallel computation should be a good candidate for other multi-objective optimization problem. Different pheromone trails and heterogeneous colonies may need to devise based on the characteristics of the problem. It should be noted that ACS algorithm reported herein is rather simple, some features normally used effectively in complex problem, such as candidate list or other local search techniques, are not incorporated in this research. There are opportunities to improve effectiveness and efficiency by considering the addition of these features to the ACS device here. Similar problem categories that may be well solved by ACS are different system structures, such as series-parallel, mixed parallel, network among others.

REFERENCES

[1] M. S. Chen, “On the computational complexity of reliability redundancy allocation in a series system,” *Operations Research Letters*, vol.11, pp. 309–315, 1992.

[2] Frank A. Tillman, Ching-Lai Hwang, Way Kuo, “Optimization Techniques for System Reliability with Redundancy - A Review”, *IEEE Transactions on Reliability*, vol. R-26, no. 3, 1977 August, pp. 148-155.

[3] Way Kuo and V. Rajendra Prasad, An annotated overview of system-reliability optimization, *IEEE Transactions on Reliability*, vol. 49, no. 2, June 2000

[4] Bellman R.E., E. Dreyfus, “Applied Dynamic Programming”, Princeton University Press, Princeton, NJ.

[5] David E. Fyffe, William W. Hines, Nam Kee Lee, “System Reliability Allocation and a Computational Algorithm”, *IEEE Transactions on Reliability*, vol. R-17, no. 2, 1968 June, pp. 64-69.

[6] Yuji Nakagawa, Satoshi Miyazaki, “Surrogate Constraints Algorithm for Reliability Optimization Problems with Two Constraints”, *IEEE Transactions on Reliability*, vol. 30, no. 2, 1981 June, pp.175-180.

[7] Ghare, P.M. and R.E. Tylar, “Optimal Redundancy for Reliability in Series Systems”, Operations research, vol.17, 1969, pp. 838-847.

[8] Bulfin, R.L. and Liu, C.Y. “Optimal allocation of redundant components for large systems”, *IEEE Transactions on Reliability*, vol. R-34, no.3, 1985, 241-247.

[9] Misra, K.B., U.Shama, “An efficient algorithm to solve integer-programming problems arising in system reliability design”, *IEEE Transactions on Reliability*, vol.40, no.1, 1991, pp303-321.

[10] Kuo W., H.Lin, Z. Xu and W. Zhang, “Reliability optimization with the Langrange multiplier and branch-and-band technique,” *IEEE Transactions on Reliability*, vol.R-36, pp.624 – 639, 1987.

[11] Tillman, F.A., C. L. Hwang and W. Kuo, “Determining component reliability and redundancy for optimum system reliability”, *IEEE Transactions on Reliability*, vol. R-26, no. 3, 1977, pp. 162-165.

[12] Hwang, C. L., F. A. Tillman and W. Kuo, “Reliability Optimization by Generalized Lagrangian-Function and Reduced-Gradient Methods”, *IEEE Transactions on*

Reliability, vol. R-28, no. 4, 1979, October, pp. 316-319.

[13] Hwang C.L. et al, "Mathematical programming with multiple objectives: a tutorial", *Computers and Operations Research*, vol.7, 1980, pp.5-31.

[14] Lieberman E. R., "Soviet multi-objective mathematical programming methods: an overview", *Management Science*, Vol.37, no.9, 1991, pp. 1147-1165.

[15] Misra, K. B., U. Shama, "An efficient approach for multi criteria redundancy optimization problems," *Microelectronics and Reliability*, vol. 31, no.2/3, 1991, pp303-321.

[16] Rao, S.S., A. K. Dhingra, "Reliability and redundancy apportionment using crisp and fuzzy multiobjective optimization approaches, "*Reliability Engineering and System Safety*", **37**(1992), 253-261.

[17] Sakawa, M., "Multiobjective optimization by the surrogate worth trade-off method", *IEEE Transaction on Reliability*, Vol. R-27, 1978 Dec, pp311-314.

[18] Inagaki, T., K. Ino, K. Ue, and H. Akashi, "Interactive optimization of system reliability under multiple objectives," *IEEE Transaction on Reliability*, Vol.R-27, no.4, pp.264-267, 1978.

[19] Goldberg, D. E., *Genetic Algorithms in Search, Optimization & Machine Learning*, 1989; Addison Wesley.

[20] Painton L. and J. Campbell, "Genetic algorithms in optimization of system reliability," *IEEE Transactions on Reliability*, vol. 44, no.2, 1995, pp. 172-178.

[21] Levitin G., A. Lisnianski, H. Ben-Haim, D. Elmakis, "Redundancy optimization for seris-parallel multi-state systems", *IEEE Transactions on Reliability*, vol. 47, no.2, 1998, pp. 165-172.

[22] David W. Coit, Alice E. Smith, "Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm", *IEEE Transactions on Reliability*, vol. 45, no. 2, 1996 June, pp. 254-260.

[23] David W. Coit, Alice E. Smith, "Penalty Guided Genetic Search for Reliability Design Optimization", *Computers and Industrial Engineering*, vol. 30, no. 4, 1996, pp. 895-904.

[24] David W. Coit, Alice E. Smith, David M. Tate, "Adaptive Penalty Methods for Genetic Optimization of Constrained Combinatorial Problems", *INFORMS Journal on Computing*, vol. 8, no. 2, 1996, pp.173-182.

[25] M. Dorigo, "Optimization, Learning and Natural Algorithms", Doctoral Dissertation, Politecnico di Milano, 1992.

[26] Dorigo M., V. Maniezzo, and A. Coloni. "Ant System: Optimization by a colony of cooperation agents". *IEEE Transactions on systems, Man and Cybernetics,* **26**:29-41, February 1996.

[27] Bland, J. A. "Optimal structural design by ant colony optimization", *Engineering Optimization*, 33, 425-443, 2001.

[28] Gianni Di Caro, Marco Dorigo, "Mobile Agents for Adaptive Routing", *Proceedings of the 31st Hawaii International Conference on System Sciences,* Big Island of Hawaii, January 6-9, 1998, pp. 74-83.

[29] Bernd Bullnheimer, Richard F. Hartl, Christine Strauss, "Applying the Ant System to the Vehicle Routing Problem", *2nd Metaheuristics International Conference (MIC-97),* Sophia-Antipolis, France, 1997, July, pp. 21-24.

[30] Mariano, C.E., E. Morales, " MOAQ an Ant Q Algorithm for multiple Objective Optimization problem", Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, Florida, Vol.1, 1999, pp894-901.

[31] Walter J. Gutjahr, "A Graph-based Ant System and its convergence", Future Generation Computing Systems, 16,873-888, June, 2000

[32] Nabil Nahas, Mustapha Nourelfath, "Ant system for reliability optimization of a series system with multiple-choice and budget constraints", Reliability Engineering and System Safety 87 (2005) 1–12

[33] Liang Y.-C. and A. E. Smith, "An Ant System approach to redundancy allocation". In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1478–1484. IEEE Press, Piscataway, NJ, 1999

[34] Liang Y.-C. and A. E. Smith, "Ant Colony Optimization for Constrained Combinatorial Problems", Proceeding of the 5th Annual International Conference on industrial Engineering Theory, Applications, and practice, Hsinchu, Taiwan, December 2000, ID 296

[35] Liang Y.-C. and A. E. Smith, "An Ant Colony Optimization Algorithm for the Redundancy Allocation Problem(RAP)", *IEEE Transactions on Reliability,* Vol. 53, No. 3, September 2004, pp417-423

[36] Dorigo M., L. M. Gambardella, "Ant Colony System: A Cooperative Learning

Approach to the Traveling Salesman Problem", *IEEE Transactions on Evolutionary Computation,* vol. 1, no. 1, 1997, pp. 53-66

[37] Stützle T.and M.Dorigo, "ACO Algorithms for the Quadratic Assignment Problem" in D. Corne, M. Dorigo, and F. Glover (eds.), *New Ideas in Optimization*, McGraw-Hill, 1999

[38] Bullnheimer, B., Richard F. Hartl, and Christine Strauss, "Applying the Ant System to the Vehicle Routing Problem", *2nd Metaheuristics International Conference (MIC-97),* Sophia-Antipolis, France, 1997, July, pp. 21-24

[39] Bauer A., B.Bullnheimer, R.F. Hartl, and C. Strauss, "An Ant Colony Optimization Approach for the Single Machine Total Tardiness Problem", Proceedings of the 1999 Congress on Evolutionary Computation, Washington, D.C., July 1999, pp.445-1450

[40] Mariano, C.E. and E. Morales, "MOAQ ant ant-Q Algorithm for Multiple Objective Optimization Problem," Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, Florida, Vol.1, 1999, pp.894-901

[41] Wordrich. M and G. Bilchev, "cooperative distributed search: The Ant's way", *Journal of control and cybernetics*, vol.26, no.3, 1997, pp413-446

[42] Dorigo M., L. M. Gambardella, "A study of Some Properties of Ant-Q", *Technical Report IRIDRA 1996-4,* 1996, Universite Libre de Bruxelle, Belgium

[43] Bullnheimer, B., R.F. Hartl, and C. Strauss, "A New Rank Based Version of the Ant System – A Computational Study", *Central European Journal for Operations Research and Economics*, vol. 7, no.1, 1999, pp.25-38

[44] Stüzle, T. and H. H. Hoos, "Max-Min Ant System", *Future Generation Computer System Journal*, vol.16, no.8, 2000, pp. 889-914

[45] Stüzle, T. and M. Dorigo, "ACO Algorithms for the Traveling Salesman Problem", in *Evolutionary Algorithms in Engineering and Computer Science*, 1999, John Wiley & Sons, pp. 163 – 183

[46] Iredi, S., D. Merkle, and M. Middendorf, "Bi-Criterion Optimization with Multi Colony Ant Algorithms", *Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization (EMO '01),* LNCS 1993, Zurich, 2001, pp. 359-372

[47] Ramalhinho, H. and D. Serra, "Adaptive Approach heuristics for the Generalized Assignment Problem," *Economic Working Paper 288*, 1998, Universitat Pompeu

Fabra, Spain

[48] Feller, W., *An Introduction to Probability Theory,* 1968, John Wiley & Sons

[49] Quy Nguyen, *Optimization of the reliability of design mechanical systems by fuzzy logic*, Master's Thesis, 1998, École de Technologie Supérieure, Université du Québec

[50] Miettinen, K., 1999, *Nonlinear Multiobjective Optimization*, Boston, Kluwer

[51] Jain, P., and Agogino, A. M., 1990, "Theory of Design: An Optimization Perspective," Mech. Mach. Theory, 25(3), pp. 287–303.

[52] Zhao, J.-H., Z. Liu and T.-M. Dao, "Reliability Optimization Using Multiobjective Ant Colony System Approaches". *Reliability Engineering and System Safety*, to appear (accepted on November, 2004).