IEMS Vol. 4, No. 1, pp. 94-101, June 2005.

A Variable Neighbourhood Descent Algorithm for the Redundancy Allocation Problem

Yun-Chia Liang[†] · **Chia-Chuan Wu**
Department of Industrial Engineering and Management, Yuan Ze University
No 135 Yuan-Tung Road, Chung-Li, Taoyuan County, Taiwan 320, R.O.C.
Tel: +886-3-4638800 ext. 2521, E-mail: ycliang@saturn.yzu.edu.tw

Abstract. This paper presents the first known application of a meta-heuristic algorithm, variable neighbourhood descent (VND), to the redundancy allocation problem (RAP). The RAP, a well-known NP-hard problem, has been the subject of much prior work, generally in a restricted form where each subsystem must consist of identical components. The newer meta-heuristic methods overcome this limitation and offer a practical way to solve large instances of the relaxed RAP where different components can be used in parallel. The variable neighbourhood descent method has not yet been used in reliability design, yet it is a method that fits perfectly in those combinatorial problems with potential neighbourhood structures, as in the case of the RAP. A variable neighbourhood descent algorithm for the RAP is developed and tested on a set of well-known benchmark problems from the literature. Results on 33 test problems ranging from less to severely constrained conditions show that the variable neighbourhood descent method provides comparable solution quality at a very moderate computational cost in comparison with the best-known heuristics. Results also indicate that the VND method performs with little variability over random number seeds.

Keywords: variable neighbourhood descent, redundancy allocation problem, series-parallel system, combinatorial optimisation

1. INTRODUCTION

The most studied configuration of the redundancy allocation problem (RAP) is a series system of s independent *k*-out-of-*n*: *G* subsystems. Because of the need for reliability and increased security requirements, a series-parallel system has been widely used. The RAP is NP-hard (Chern 1992) and has been studied in many forms as summarized in Tillman *et al*., (1977a), and by Kuo and Prasad (2000).

As shown in Figure 1, a *k*-out-of-*n*: *G* subsystem *i* is functioning properly if at least k_i of its n_i components are operational. In the formulation of a series-parallel system problem, for each subsystem, multiple component choices are used in parallel, and each subsystem may have different component selections. Thus, the RAP can be formulated to select the optimal combination of components and redundancy levels to meet system level constraints, cost of *C* and weight of *W* while maximizing system reliability. It is assumed that system weight and system cost are represented by linear combinations of component weight and cost.

$$\text{Max} \quad R = \prod_{i=1}^{s} R_i(\mathbf{y}_i \mid k_i) \tag{1}$$

Subject to

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

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

In addition, if the maximum number of components allowed in parallel is pre-determined, the following constraint is added:

$$k_i \le \sum_{j=1}^{a_i} y_{ij} \le n_{\max} \qquad \forall i = 1, 2, \ldots, s. \tag{4}$$

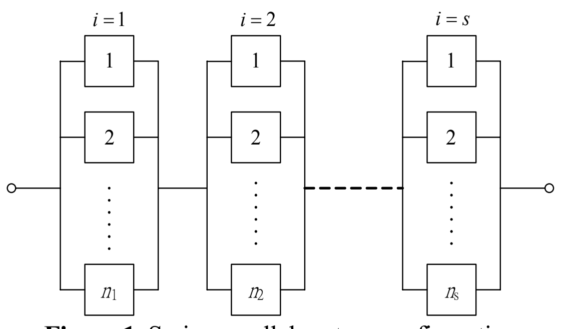

Figure 1. Series-parallel system configuration

† : Corresponding Author

1.1 Notations and Assumptions

k	minimum number of components required to function as a pure parallel system
n	total number of components used in a pure parallel system
k-*out-of-*n: G	a system that functions when at least k of its n components function
R	overall reliability of the series-parallel system
C	system cost constraint
W	system weight constraint
S	number of subsystems
i	index for subsystem, $i = 1,\ldots, s$
j	index for components in a subsystem
a_i	number of available component choices for subsystem i
r_{ij}	reliability of component j available for subsystem i
c_{ij}	cost of component j available for subsystem i
w_{ij}	weight of component j available for subsystem i
y_{ij}	quantity of component j used in subsystem i
$\mathbf{y}_i$	$(y_{i1},\ldots,y_{ia_i})$
n_i	$=\sum_{j=1}^{a_i} y_{ij}$, total number of components used in subsystem i
$n_{\max}$	maximum number of components that can be in parallel (user assigned)
k_i	minimum number of components in parallel required for subsystem i to function
$R_i(y_i \mid k_i)$	reliability of subsystem i, given k_i
$C_i(y_i)$	total cost of subsystem i
$W_i(y_i)$	total weight of subsystem i
R_u	un-penalized system reliability of solution u
R_{up}	penalized system reliability of solution u
C_u	total system cost of solution u
W_u	total system weight of solution u
γ_C	amplification parameter of cost constraint in the penalty function
γ_W	amplification parameter of weight constraint in the penalty function
$l_{\max}$	number of neighbourhood structures
l	index for neighbourhood structures
N_l	the l^{th} neighbourhood structure in the search sequence
y	current solution
y'	the best neighbouring solution of y in a neighbourhood structure
e	constant that controls the lower bound of number of components used in parallel
f	constant that controls the upper bound of number of components used in parallel

Typical assumptions are considered as follows:

- The state of the components and the system either function or fail.
- Failed components do not damage the system, and are not repaired.
- The failure of a component does not lead to other components failing.
- Components are active redundant, i.e., the failure rates of components when not in use are the same as when in use.
- Component reliability, weight and cost are known and deterministic.
- The supply of components is unlimited.

1.2 Literature Review

Exact methods of the RAP include dynamic programmming (Bellman and Dreyfus 1958, Fyffe *et al.* 1968, Nakagawa and Miyazaki 1981, Li 1996), integer programming (Ghare and Taylor 1969, Bulfin and Liu 1985, Misra and Sharma 1991, Coit and Liu 2000), and mixed-integer and nonlinear programming (Tillman et al. 1977b). Since exact methods in practice are limited to the increase in problem size, meta-heuristics have become a popular alternative to exact methods. Meta-heuristic approaches to the RAP vary among Genetic Algorithm (GA) (Coit and Smith 1996a&b, Levitin *et al.* 1998), Tabu Search (TS) (Huang *et al.* 2002, Kulturel-Konak *et al.* 2003), Ant Colony Optimization (ACO) (Liang and Smith 1999, Huang *et al.* 2002, Liang 2001, Liang and Smith 2004), hybrid Neural Network (NN) and GA (Coit and Smith 1996c), and hybrid ACO with TS (Huang 2003). In particular, Levitin *et al.* (1998) generalize a redundancy allocation problem to multi-state systems, where the system and its components have a range of performance levels-from perfectly functioning to complete failure. Levitin's paper implements a universal moment genera-ting function to estimate system performance (capacity or operation time), and a GA is employed as the optimization technique. Furthermore, considering the problem type for maximizing system reliability, past studies such as TS proposed by Kulturel-Konak *et al.* (2003), GA de-veloped by Coit and Smith (1996a), and ACO by Liang and Smith (2001, 2004) have provided comparable results over a set of 14-subsystem benchmark problems. Huang *et al.* (2002, 2003) proposed ACO, TS, and hybrid ACO/TS algorithms respectively to solve the system cost minimization RAP.

Our study considers one of the latest meta-heuristics, Variable Neighbourhood Descent (VND), to solve the system reliability maximization RAP. A variation of the variable neighbourhood search (VNS) method was introduced first by Mladenović (1995). The VND method explores search space based on the systematic change of neighbourhoods within the process and has been successfully applied to diverse combinatorial optimisation problems, e.g. multi-source problem (Brimberg *et al.* 2000), single machine scheduling (Besten and Stützle 2001), arc routing problem (Hertz and Mittaz 2001), clustering (Hansen and Mladenović 2002), minimum spanning tree problem (Ribeiro and Souza 2002), and phylogeny problem (Ribeiro and Vianna 2005). Because of the proper neighbourhood structure of the RAP and lack of dominant solution techniques, it is a good candidate for other meta-heuristic approaches including the focus of this paper, VND.

This paper is organized as follows. Section 2 introduces the variable neighbourhood descent algorithm for RAP, and section 3 provides our discussion on computational results. Finally, section 4 contains concluding remarks and suggestions for future research.

2. VARIABLE NEIGHBOURHOOD DESCENT ALGORITHM

Hansen and Mladenović (2003) provide a detailed summary of the state-of-the-art development of VNS and its variations. They describe VNS-type algorithms as meta-heuristics, and employ a set of neighbourhood search methods to find the local optimum in each neighbourhood iteratively and hopefully to reach the global optimum at the end. If the change of neighbour-hoods is performed in a deterministic way, such a method is called VND. The VND algorithm for RAP (denoted as VND_RAP) is discussed below.

At the initialization step, a set of neighbourhood structures (N_l; $l = 1,...,l_{\max}$) and the sequence of their implementation are determined. Then a feasible initial solution is generated and set as the current solution (y). In the main search loop, starting from the first neighbourhood, *i.e.*, N_1, *a* complete neighbourhood search is performed. If the best neighbouring solution (y') in this neighbourhood is better than the current solution (y), y is replaced by y' and l is reset to 1, *i.e.*, start from the first neighbourhood again with the updated y. Otherwise, start the consecutive neighbourhood search with y. The process continues until all neighbourhoods are visited and no further improvement can be obtained to the current solution. The VND_RAP procedure can be formalized in the following flow chart:

Three key factors underlie the proposed VND_RAP algorithm - how to generate a proper initial solution, how to define a set of neighbourhood structures, and how to design a penalty function to evaluate infeasible solutions are discussed in the following sections.

2.1 Initial Solution Generation

To generate a feasible initial solution, s integers between k_i + e and $n_{\max}$ - f (inclusive) are randomly selected to represent the number of components in parallel (n_i) for each subsystem. e and f denote the constants that control the range of component selection. Then, the n_i components are chosen under a uniform distribution to the a_i different types. Where a feasible solution is calculated, the initial solution is found; otherwise, the process is repeated until a feasible initial solution is obtained.

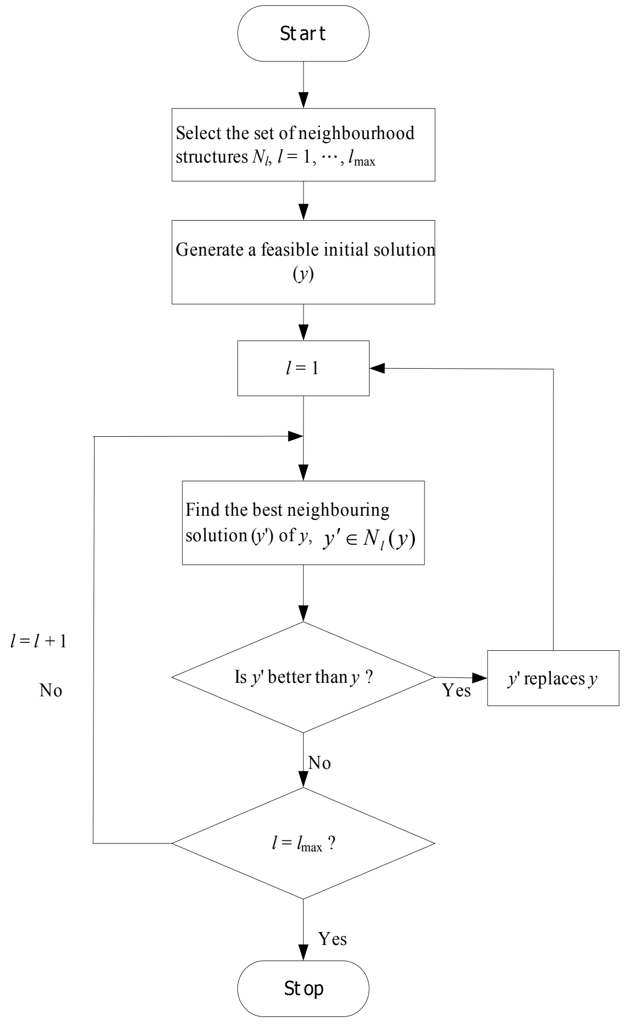
Start
Select the set of neighbourhood structures N_l, l = 1, ⋯, $l_{\max}$
Generate a feasible initial solution (y)
l = 1
Find the best neighbouring solution (y') of y, $y' \in N_l(y)$
Is y' better than y ?
Yes
y' replaces y
No
$l = l_{\max}$?
$l = l + 1$
No
Yes
Stop

Figure 2. Flow chart of VND_RAP algorithm

2.2 Neighbourhood Structure

This section depicts four kinds of neighbourhood structures and illustrates with an example, respectively:

Type 1 structure: This structure considers changes on only one subsystem. Simultaneously replace one type of component with a different type within the same subsystem, *i.e.*, $y_{ij} \rightarrow y_{ij} + 1$ and $y_{im} \rightarrow y_{im}$ - 1 for $j \neq m$. All possibilities are enumerated and all subsystems are considered. For example, if there are three available components (I, II, and III) in a subsystem, and the current component allocation is as in Figure 3(a). Figures 3(b) to 3(e) show four combinations of neighbouring solutions generated accordingly and the shaded components are newly added to replace the existing ones.

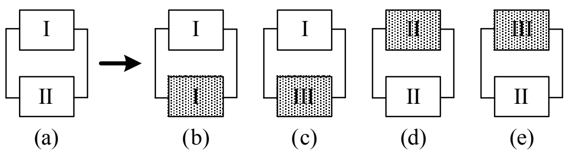

Figure 3. An example of neighbourhood type 1

Type 2 structure: They are the same as in a Type 1 structure and changes the number of a particular component type by either adding or subtracting one, *i.e.*, $y_{ij} \rightarrow y_{ij} + 1$ or $y_{ij} \rightarrow y_{ij} - 1$. Each component available in a subsystem is independently considered. For example, there are three component types (I, II, and III) available in a particular subsystem and Figure 4(a) represents the current allocation with one Type I component and two Type II components. Then, a total of five possible combinations of neighbouring solutions are derived as Figures 4(b) to 4(f), where shaded boxes represent the added components, and the removed components are marked by the dashed box.

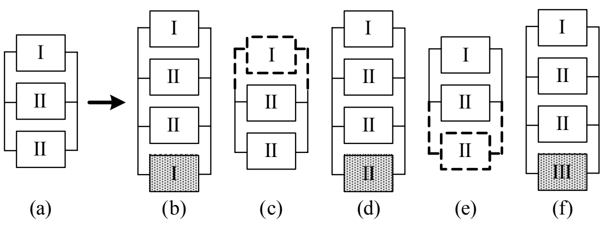

Figure 4. An example of neighbourhood type 2

Type 3 structure: This structure considers changes on two subsystems at a time. Hence, simultaneously add one component to a certain subsystem and delete one component in another subsystem, *i.e.*, $y_{ij} \rightarrow y_{ij} + 1$ and $y_{om} \rightarrow y_{om} - 1$ for $i \neq o$. All possibilities are enumerated and all subsystems are considered. For instance, the current allocation is demonstrated in Figure 5(a). One possible neighbouring solution is to add one Component III in subsystem 1 and delete Component I from subsystem i; the result is in Figure 5(b) where the shaded box is the added component without replacing any existing one, and the dashed box is the eliminated one.

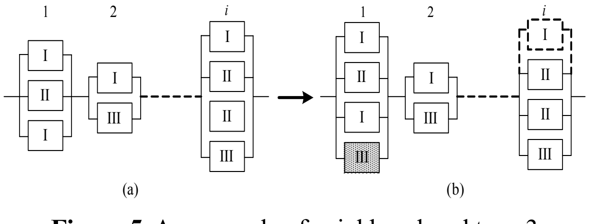

Figure 5. An example of neighbourhood type 3

Type 4 structure: They are the same as in a Type 3 structure. This neighbourhood extends the Type 1 neighbourhood above to exchange components in two of many subsystems simultaneously, *i.e.*, $(y_{ij} \rightarrow y_{ij} + 1$ and $y_{im} \rightarrow y_{im} - 1) \cup (y_{oj} \rightarrow y_{oj} + 1$ and $y_{om} \rightarrow y_{om} - 1)$ for $j \neq m$ and $i \neq o$. An example shown in Figure 6 assumes three component options (denoted by I, II, and III) available in each subsystem. The example of Figure 6(a) shows the current status. In subsystem 1, a Component II replaces one of Component Is. Also, in subsystem i a Component III substitutes one of Component IIs. Figure 6(b) shows the result of these two subsystems where the new components are shaded.

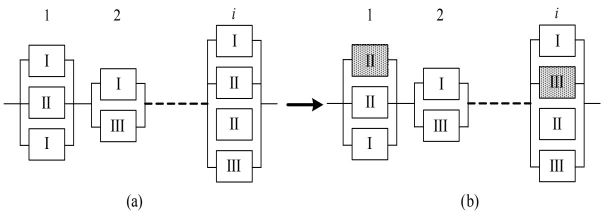

Figure 6. An example of neighbourhood type 4

2.3 Penalty Function

Coit and Smith (1996b) suggest the use of a penalty function to evaluate the objective function under a constrained problem. An appropriate-designed penalty function will encourage the algorithm to explore the feasible region and infeasible region near the border of the feasible area, and discourage, but permit, further search into the infeasible region. After generating all neighbouring solutions, the algorithm uses a penalty function for infeasible solutions (Liang and Smith 2004):

$$R_{up} = R_u \times \left(\frac{C}{C_u} \right)^{\gamma_C} \times \left(\frac{W}{W_u} \right)^{\gamma_W} \tag{5}$$

where C_u and W_u are total system cost and weight of solution u respectively. R_u is the unpenalized reliability of solution u calculated using equation (1). Then, the penalized reliability R_{up} for systems that exceed cost constraint C and/or weight constraint W is calculated by multiplying the un-penalized objective with two penalty factors $\left(C / C_u \right)^{\gamma_C}$ and $\left(W / W_u \right)^{\gamma_W}$ where the exponents γ_C and γ_W are preset amplification parameters.

3. COMPUTATIONAL RESULTS

The VND_RAP is coded in Borland C++ Builder 5.0 and run with an Intel Pentium IV 2.8GHz PC at 512 MB

Table 1. Component data for test problems (Fyffe *et al.*, 1968)

Sub-system (i)	Component Alternatives (j)											
	1			2			3			4		
	r_{i1}	c_{i1}	w_{i1}	r_{i2}	c_{i2}	w_{i2}	r_{i3}	c_{i3}	w_{i3}	r_{i4}	c_{i4}	w_{i4}
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	n/a	n/a	n/a
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	n/a	n/a	n/a
5	0.94	2	4	0.93	2	3	0.95	3	5	n/a	n/a	n/a
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	n/a	n/a	n/a
8	0.81	3	4	0.90	5	7	0.91	6	6	n/a	n/a	n/a
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	n/a	n/a	n/a
11	0.94	3	5	0.95	4	6	0.96	5	6	n/a	n/a	n/a
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	n/a	n/a	n/a
14	0.90	4	6	0.92	4	7	0.95	5	6	0.99	6	9

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 them with results in the literature.

The 33 variations of the Fyffe *et al.* problem (1968) as devised by Nakagawa & Miyazaki (1981) were used to test the performance of VND_RAP. The component options for the benchmark problems are listed in Table 1. This problem sets C = 130 and W incrementally decreased from 191 to 159. In Fyffe *et al.* (1968) and Nakagawa & Miyazaki (1981), the optimization approaches require only identical components to be placed in redundancy; however, for the VND_RAP approach, different types are allowed to reside in parallel (assuming that values of $n_{\max}$ = 8 and k_i = 1 for all subsystems). The parameters of the penalty function are set as $\gamma_c = 1$ and $\gamma_W = 2$. Ten runs were made using different random number seeds for each problem instance.

The following two settings play key roles in the VND_RAP algorithm: the number of initial components in each subsystem and the sequence of the neighbourhood structures. Therefore, details of these two factors are investigated in sections 3.1 and 3.2. In addition, considering prior RAP work where component mixing was allowed on the heuristic benchmarks, the study chose GA of Coit and Smith (1996a), TS of Kulturel-Konak *et al.* (2003), and ACO of Liang and Smith (2004) for comparison and will be discussed in section 3.3.

3.1 Number of Initial Components

The generation of an initial solution is controlled by a range between $k_i + e$ and $n_{\max} - f$ (inclusive) determining the number of components in parallel in each subsystem. In order to obtain a feasible initial solution close to the border between the feasible region and the infeasible region, an investigation is conducted. The algorithm considers both ($e = 1, f = 4$) and ($e = 1, f = 5$); in other words, 2 to 4 components and 2 to 3 components are selected for each subsystem. Figure 7 indicates the maximum reliability over ten runs in 33 instances. Under the least (W = 191~181) and moderate (W = 180~170) constrained problems, the option of 2~4 components performs equal or better than that of 2~3 components in 14 out of 22 instances. When the weight constraint becomes stricter (W = 169~159), the option of 2~3 components outperforms the 2~4 components one in 7 out of 11 instances. Then, with the decrease of the W value, Table 2 shows the computational expense of the 2~4 components option grows tremendously. For the most constrained 11 instances, the average CPU time increases to approximately 96 seconds. However, the 2~3 components option performs with much less computationally effort. Our results also show whether the constraints are loose or strict, the average CPU time always appears lower than 1 second. To balance solution quality and computational expense, the 2~4 components option is adopted for the least and moderate constrained instances, i.e., W = 191~170, and remaining instances use the 2~3 components option.

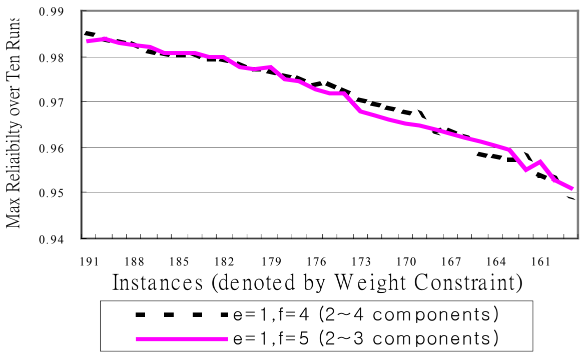

Figure 7. The effect of number of initial components on system reliability

Table 2. The effect of number of initial components on average CPU time (in seconds)

	191~181	180~170	169~159
2~4	0.339	1.443	95.912
2~3	0.291	0.266	0.582

3.2 The Sequence of Neighbourhood Structures

Hansen and Mladenović (2003) indicate that the key to successfully employing the VND consists of the proper definition of neighbourhood structures, and the sequence of neighbourhood structures, etc. Four types of neighbourhood structures are proposed in section 2.2, so the number of possible search sequences is 4! = 24. Figure 8 shows the average system reliability over 10 runs in each possible sequence of neighbourhood structures. The instances are divided into three groups – 191~181, 180~170, and 169~159 depending on the severity of the weight constraint. The sequence of 2-3-1-4 outperforms other sequences in all three groups of instances. In addition, when starting with a Type 2 neighbourhood, the average performance is better than starting with other structures.

Table 3 summarizes the average performance (denoted by $\overline{R}_{avg}$), the best performance (denoted by $\overline{R}_{max}$), and theaverage standard deviation (denoted by Stdev) over 10 runs in all 33 instances for each possible se-quence. It again confirms that the sequence of Type 2-Type 3-Type 1-Type 4 performs the best in all measures. That is this sequence not only provides the best solution quality but also is the most robust to the random number seed. Therefore, in next section, this sequence is used to compare other best-known heuristics in the literature.

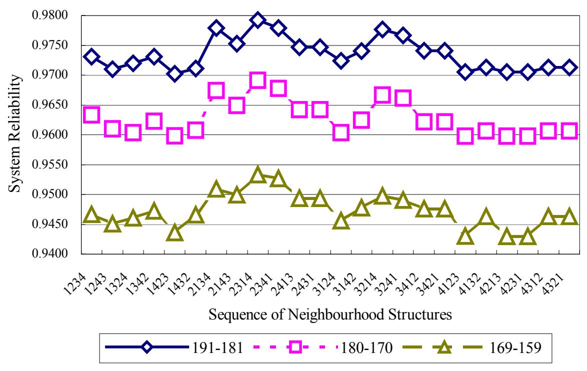

Figure 8. The effect of sequence of neighbourhood structures on average system reliability

3.3 Comparison with Other Metaheuristics

Considering prior RAP works where component mixing was allowed on the heuristic benchmarks, the three best-known metaheuristics - GA of Coit and Smith (1996a), TS of Kulturel-Konak *et al.* (2003), and ACO of Liang and Smith (2004) were chosen for comparison. Each algorithm was run 10 times using different random number seeds for each instance. Percentages of the deviation between VND_RAP and other meta-heuristics, GA, ACO, and TS are displayed in Figure 9. The performance measure is defined as ((other methods – VND_RAP) / VND_RAP) × 100%. The best solution over 10 runs was used for comparison. The deviations for the less and moderate constrained instances (the first 22) scatter between 0.15% and 0.35%, and when the problems become more constrained (the last 11), the deviations fluctuate between 0.45% and 0.01%. The

Table 3. The effect of sequence of neighbourhood structures on $\overline{R}_{avg}$, $\overline{R}_{max}$, and Stdev

Sequence	$\overline{R}_{avg}$	$\overline{R}_{max}$	Stdev	Sequence	$\overline{R}_{avg}$	$\overline{R}_{max}$	Stdev
1-2-3-4	0.9610	0.9669	0.0045	3-1-2-4	0.9594	0.9657	0.0042
1-2-4-3	0.9590	0.9652	0.0046	3-1-4-2	0.9614	0.9668	0.0037
1-3-2-4	0.9595	0.9654	0.0041	3-2-1-4	0.9647	0.9698	0.0044
1-3-4-2	0.9609	0.9665	0.0039	3-2-4-1	0.9640	0.9694	0.0044
1-4-2-3	0.9579	0.9649	0.0050	3-4-1-2	0.9613	0.9664	0.0037
1-4-3-2	0.9595	0.9656	0.0039	3-4-2-1	0.9613	0.9663	0.0037
2-1-3-4	0.9654	0.9701	0.0038	4-1-2-3	0.9578	0.9648	0.0049
2-1-4-3	0.9634	0.9695	0.0043	4-1-3-2	0.9594	0.9656	0.0039
2-3-1-4	**0.9672**	**0.9712**	**0.0030**	4-2-1-3	0.9578	0.9648	0.0049
2-3-4-1	0.9661	0.9702	0.0031	4-2-3-1	0.9578	0.9648	0.0049
2-4-1-3	0.9627	0.9687	0.0040	4-3-1-2	0.9594	0.9656	0.0039
2-4-3-1	0.9628	0.9688	0.0040	4-3-2-1	0.9594	0.9656	0.0039

average percentage of the deviation VND_RAP over all 33 instances was less than 0.3% compared with other three methods. In addition, the average standard deviation of VND_RAP over 330 runs (33 instances with 10 runs each) was 0.003. Thus, VND_RAP was able to find comparable solutions to the best-known methods in the literature and was robust to random number seeds.

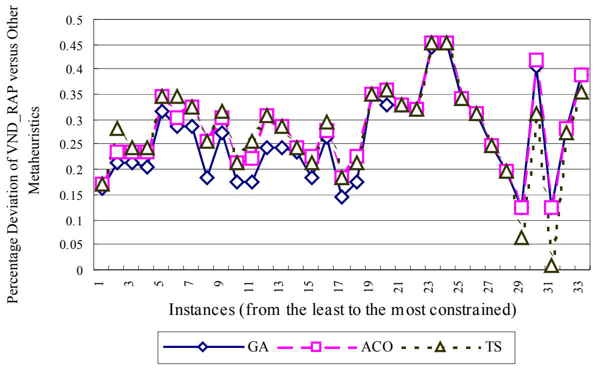

Figure 9. Comparisons of VND_RAP versus GA, ACO, and TS

It is difficult to make a precise comparison on computational expense. CPU seconds vary according to hardware, software and programmer's coding skill. Therefore, the number of solutions generated may provide a better idea on how efficient an algorithm performs. The number of solutions generated in GA (Coit and Smith 1996b) was 48,040 (a population size of 40 and stopped after 1200 iterations), the ACO algorithm in Liang and Smith (2004) needed about 100,000 evaluations (a colony size of 100 with up to 1000 iterations), and TS in Kulturel-Konak *et al.* (2003) evaluated an average of 350,000 solutions. The number of solutions searched in VND_RAP was approximately 49,000 on average. That is, VND_RAP required only a similar number of evaluations to GA, less than half of the ones in ACO, and 1/7 of TS to get the comparable performance. In addition, the average CPU time of VND_RAP was 0.73 seconds that was considered as a reasonable time in solving such a large system.

4. CONCLUSIONS

This paper uses a variable neighbourhood descent meta-heuristic method to solve the series-parallel redundancy allocation problem. The RAP is a well-known NP-hard problem generally in a restricted form where each subsystem must consist of identical components in parallel to make computations tractable. The newer meta-heuristic methods overcome this limitation and offer a practical way to solve large instances of the relaxed RAP where different components can be placed in parallel. Given the well-structured neighbourhood of the RAP, variable neighbourhood descent method that has not been used in reliability design yet is likely to be more effective and more efficient than one that does not. A VND_RAP algorithm is proposed and tested on the most well known benchmark problems from the literature. It has been shown that VND_RAP performs comparably well to other meta-heuristics in solution quality and very efficiently in computational expense consideration. Therefore, VND and its variations seem very promising for other NP-hard reliability design problems such as those found in networks and complex structures.

ACKNOWLEDGEMENT

The authors would like thank Dr. Shu-Kai S. Fan for his precious comments on this research.

REFERENCES

Bellman, R. and Dreyfus, S. (1958), Dynamic Programming and the Reliability of Multicomponent Devices, *Operations Research*, 6, 200-206.

Besten, M. D. and Stützle, T. (2001), Neighborhoods Revisited: an Experimental Investigation into the Effectiveness of Variable Neighborhood Descent for Scheduling, *Proceedings of the 4th Metaheuristics International Conference*, Proto, Portugal, 545-549.

Brimberg, J., P. Hansen, Mladenović, N. and Taillard, É. (2000), Improvements and Comparison of Heuristics for Solving the Multisource Weber Problem, *Operations Research*, 48(3), 444-460.

Bulfin, R. L. and Liu, C. Y. (1985), Optimal Allocation of Redundant Components for Large Systems, *IEEE Transactions on Reliability*, R-34(3), 241-247.

Chern, M. S. (1992), On the Computational Complexity of Reliability Redundancy Allocation in a Series System, *Operations Research Letters*, 11, 309-315.

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

Coit, D. W. and Smith, A. E. (1996a), Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm, *IEEE Transactions on Reliability*, 45(2), 254-260.

Coit, D. W. and Smith, A. E. (1996b), Penalty Guided Genetic Search for Reliability Design Optimization, *Computers & Industrial Engineering*, 30(4), 895-904.

Coit, D. W. and Smith, A. E. (1996c), Solving the Redundancy Allocation Problem Using a Combined Neural Network/Genetic Algorithm Approach, *Computers & Operations Research*, 23(6), 515-526.

Fyffe, D. E., Hines, W. W. and Lee, N. K. (1968), System Reliability Allocation and a Computational Algorithm, *IEEE Transactions on Reliability*, R-17(2), 64-69.

Ghare, P. M. and Taylor, R. E. (1969), Optimal Redundancy for Reliability in Series Systems, *Operations Research*, 17, 838-847.

Hansen, P. and Mladenović, N. (2002), J-Means: A New Local Search Heuristic for Minimum Sum-of-Squares Clustering, *Pattern Recognition*, 34, 405-413.

Hansen, P. and Mladenović, N. (2003), Variable Neighborhood Search, In F. W. Glover and G. A. Kochenberger (eds.), *Handbook of Metaheuristics*, Kluwer Academic Publisher, 145-184.

Hertz, A. and Mittaz, M. (2001), A Variable Neighborhood Descent Algorithm for the Undirected Capacitated Arc Routing Problem, *Transportation Science*, 35, 425-434.

Huang, Y.-C. (2003), *Optimization of the Series-Parallel System with the Redundancy Allocation Problem Using a Hybrid Ant Colony Algorithm*, Master Thesis, Yuan Ze University, Taiwan, R.O.C.

Huang, Y.-C., Her, Z.-S. and Liang, Y.-C. (2002), Redundancy Allocation Using Meta-Heuristics, *Proceedings of the 4th Asia-Pacific Conference on Industrial Engineering and Management System* (*APIEMS* 2002), Taipei, December 2002, 1758-1761.

Kulturel-Konak, S., Coit, D. W. and Smith A. E. (2003), Efficiently Solving the Redundancy Allocation Problem Using Tabu Search, *IIE Transactions*, 35(6), 515-526.

Kuo, W. and Prasad, V. R. (2000), An Annotated Overview of System-reliability Optimization, *IEEE Transactions on Reliability*, 49(2), 176-187.

Levitin, G., Lisnianski, A., Ben-Haim, H. and Elmakis, D. (1998), Redundancy Optimization for Series-Parallel Multi-State Systems, *IEEE Transactions on Reliability*, 47(2), 165-172.

Li, J. (1996), A Bound Dynamic Programming for Solving Reliability Optimization, *Microelectronic Reliability*, 36(10), 1515-1520.

Liang, Y.-C. (2001), *Ant Colony Optimization Approach to Combinatorial Problems*, Ph.D. Dissertation, Auburn University, USA.

Liang, Y.-C. and Smith, A. E. (1999), An Ant System Approach to Redundancy Allocation, *Proceedings of the 1999 Congress on Evolutionary Computation*, Washington D.C., July 1999, 1478-1484.

Liang, Y.-C. and Smith, A. E. (2004), An Ant Colony Optimization Algorithm for the Redundancy Allocation Problem (RAP), *IEEE Transactions on Reliability*, 53(3), 417-423.

Misra, K. B. and Sharma, U. (1991), An Efficient Algorithm to Solve Integer-Programming Problems Arising in System-Reliability Design, *IEEE Transactions on Reliability*, 40(1), 81-91.

Mladenović, N. (1995), A Variable Neighborhood Algorithm - A New Metaheuristic for Combinatorial Optimization, Abstracts of papers presented at Optimization Days, Montréal, 112.

Nakagawa, Y. and Miyazaki, S. (1981), Surrogate Constraints Algorithm for Reliability Optimization Problems with Two Constraints, *IEEE Transactions on Reliability*, R-30(2), 175-180.

Ribeiro, C. C. and Souza, M. C. (2002), Variable Neighborhood Search for the Degree-Constrained Minimum Spanning Tree Problem, *Discrete Applied Mathematics*, 118, 43-54.

Ribeiro, C. C. and Vianna, D. S. (2005), A GRASP/VND Heuristic for the Phylogeny Problem Using a New Neighborhood Structure, *International Transactions in Operational Research*, 12, 1-14.

Tillman, F. A., Hwang, C. L. and Kuo, W. (1977a), Optimization Techniques for System Reliability with Redundancy - A Review, *IEEE Transactions on Reliability*, R-26(3), 148-155.

Tillman, F. A., Hwang, C. L. and Kuo, W. (1977b), Determining Component Reliability and Redundancy for Optimum System Reliability, *IEEE Transactions on Reliability*, R-26(3), 162-165.