Link enhancement using constrained range and reduced candidate set searches: a revision

Ruey-yih Lin and Chyan Yang

This paper revises the CRCS algorithm proposed by Yang[1]. In the constrained range search, we use the cost and performance of candidate links to obtain the exact link range of the optimal solution. Moreover, we use the concept of variable dominance to reduce the number of candidate links without loss of optimality. A comparison of results obtained using Yang's and the revised CRCS algorithm shows that the revised CRCS algorithm is superior to Yang's original version.

Keywords: candidate set search, constrained range search, variable dominance

In a recent paper in this journal, Yang[1] proposed a heuristic algorithm that uses constrained range and reduced candidate set search (CRCS) to reduce computation time while obtaining a near-optimal solution. The key finding in Yang's paper is that the CRCS algorithm is justified, since the algorithm takes only a non-exponential time to compute and has a high probability of reaching optimality.

However, Yang's work raises two problems that need to be investigated. First, the range of the constrained range search in Yang's paper, which uses the costs of candidate links to obtain the range (or number of links) of the optimum solution, is very wide, so a further 'squeeze' is required. Second, since Yang's method chooses a candidate set through a linear search algorithm, it is possible that optimality will be lost, i.e. the reduced candidate set in Yang's method could rule out a link which is in the optimal solution set.

The purpose of this paper is to revise the CRCS algorithm. Two major improvements are proposed: first, by combining information on the cost and performance of candidate links, we obtain the exact number of links in one step without compromising optimality and without requiring further 'squeeze'; second, instead of using a linear search algorithm, we use the concept of *variable dominance* to reduce the candidate set and ensure that the optimal solution is in the reduced set. Further, we include the dominance of variables in the computation to increase searching efficiency.

The rest of this paper is organized as follows. The revised CRCS algorithm is presented in the next section; the following section compares the results obtained using Yang's CRCS algorithm with those obtained using the revised algorithm. Conclusions and the limitations of CRCS algorithms are discussed in the final section.

REVISION OF THE CRCS ALGORITHM

Two major improvements can be made to Yang's CRCS algorithm. The first is to tighten the feasible space, which is determined by the available budget and the given cost of the links. The second is to use the concept of variable dominance to retain (or rule out) links that are definitely (or definitely not) in the optimal solution set without losing optimality. We shall call the revised CRCS method *CRCS'*. We explain this search method as follows.

New constrained range search

First, we describe the method of constraining the range in the *CRCS'* algorithm. Given a budget B and the cost C_i and performance P_i of each candidate link i, we may squeeze the number of links within a constrained range without compromising optimality, hence reducing the computational cost tremendously.

The revised constrained range search algorithm can tighten the number of optimal links exactly as is done in Yang's examples. First,

Institute of Management Science & Institution of Information Management, National Chiao-Tung University, Hsinchu, Taiwan, ROC (email: cyang@cc.nctu.edu.tw)
Paper received: 3 October 1993

we sort the cost of all links in ascending order and add up the cost of each link from the lowest cost link up until the budget is exhausted. Then we obtain the number of links under this cumulative cost, and take this number as an upper bound on the count of the optimal solution set (UL), since this cumulative cost is the cost of the maximum number of links we can use. Second, we sort the performance of all links in descending order. These sorted links are selected in order and their costs are added until the budget is exhausted. Then we obtain the number of links under this cumulative cost and take this number as a lower bound on the count of the optimal solution set (LL), since this cumulative performance is the best we can achieve with this number of links. Then, combining these two steps, we tighten the number of links in the optimal solution set (K), which should lie in the interval $LL \leqslant K \leqslant UL$. Notice that, when the distribution (or variation) of the data set is not extreme, then the upper bound UL will be close to the lower bound LL. If the constrained range search method can tighten the number of links so that the inequality $LL \leqslant K \leqslant LL + 1 = UL$ is satisfied, then the reduced candidate set search in the next step will be easier to carry out. Fortunately, in Yang's paper, the variance in all four data sets is not great, so using our constrained range search method, it is easy to obtain $LL \leqslant K \leqslant LL + 1 = UL$. The number of possible combinations we need to try is reduced to $C(N, UL)$, where N is the total number of links. Moreover, since in our constrained range search method we do not compromise optimality, if K is equal to the lower bound LL, the optimal solution (maximum performance) is obtained simultaneously in this step.

New reduced candidate set search

After the number of links is determined, we employ the concept of variable dominance in conjunction with the upper bound UL to retain (or drop) links that are necessarily in (or not in) the optimal solution set. This step helps to reduce the computation time significantly.

First, we define the concept of variable dominance:

$$x_i \text{ dominates } x_j \quad \text{iff } P_i \geqslant P_j \text{ and } C_i \leqslant C_j$$

where x_i is a binary decision variable and $x_i = 1$ ($x_i = 0$) of link i is selected (not selected). P_i and C_i are the performance and cost of link i, respectively.

We have an obvious property of dominance:

$$\text{if } x_i \text{ dominates } x_j \quad \text{then } x_i \geqslant x_j$$

Now, we define binary variable $D(i,j)$. Let $D(i,j) = 1$ if link i dominates (or is superior to) link j, and 0 otherwise. By pairwise comparison of every link, we define $WIN(i) = \sum_{j \neq i} D(i,j)$ and $LOSE(i) = \sum_{j \neq i} D(j,i)$ as the total number of links i which dominate or are dominated by all other links, respectively. Notice that in the constrained range search we find the exact number of links ($K = UL$) in the optimal solution set, and the total number of links is N. Then link i must be retained in the optimal solution set if $WIN(i) \geqslant N - UL$. This is because if link i is not in the optimal solution set, i.e. $x_i = 0$, then the decision variables of the other $WIN(i) \geqslant N - UL$ links dominated by link i will be forced to be zero. Then there exist only $UL - 1$ remaining links in the optimal solution set, which contradicts the result of the constrained range search. Furthermore, a link i is ruled out in the optimal solution set if $LOSE(i) \geqslant UL$, since if link i is in the optimal solution set ($x_i = 1$), then the other UL links that dominate link i should also be in the optimal solution set, and the budget constraint will not hold when $UL + 1$ links are in the solution set.

Notice that when the resources (or budget) are more limited, then a link dominated by other links has a higher probability of not being retained in the candidate set, since the range UL of the optimal solution will become smaller and the inequality $LOSE(i) \geqslant UL$ will be easier to satisfy. On the other hand, when resources are sufficient, the upper bound UL of the optimal solution is closer to the total number of links N, i.e. the inequality $WIN(i) \geqslant N - UL$ is easier to satisfy, so a link that dominates other links will have a higher probability of being retained in the optimal solution set.

With the revised $CRCS'$ algorithm, we discard a link when the total number of links that dominate it is greater than or equal to the upper bound, and we retain a link when the total number of links it dominates is greater than or equal to the total number of links minus the upper bound. In other words, we reduce the feasible space by tightening the candidate set. In this reduced candidate set search, we use only an adding and sorting procedure, whereas in Yang's paper a division or ratio scale is used. So our revised CRCS method not only saves much computational time, but also retains all the information on optimality, i.e. our $CRCS'$ algorithm is thus superior to Yang's algorithm. It is worth mentioning that Walukiewicz[2] showed that including the dominance of variables in a reduction method can increase the method's efficiency[3]. After we obtain the reduced candidate set, we can add the remaining dominance inequalities as new constraints to make the optimality computation algorithm more efficient.

Performance of revised CRCS

In this section, we use the examples presented in Yang's paper to show that the revised CRCS algorithm is more robust than Yang's original algorithm. We apply the revised CRCS algorithm to the problem of *Table 1*. For example, by using the new constrained range search in the case of insufficient resources with total number of links $N = 20$ and budget of 7000, we can tighten the number of links between $3 \leqslant K \leqslant 4$, and since the maximum total profit is 14193 when $K = 3$, we need to

Table 1 Data set for $N = 20$, $B = 7000$

Ascending sorting by cost			*Descending sorting by performance*		
Link No.	*Cost*	*Performance*	*Link No.*	*Cost*	*Performance*
3	1246	3819	6	2568	5276
7	1508	3859	8	1608	4477
4	1529	2310	13	2184	4440
20	1578	3484	17	1883	4368
8	1608	4477	16	2289	4224
18	1682	1922	1	1833	4140
9	1691	3269	7	1508	3859
19	1711	3844	19	1711	3844
2	1754	3506	3	1246	3819
1	1833	4140	10	2112	3807
11	1840	3661	11	1840	3661
17	1883	4368	15	2254	3643
12	1960	3560	12	1960	3560
5	2034	3370	2	1754	3506
10	2112	3807	20	1578	3484
13	2184	4440	5	2034	3370
15	2254	3643	9	1691	3269
16	2289	4224	14	2549	2899
14	2549	2899	4	1529	2310
6	2568	5276	18	1682	1922
$UL = 4$	$TC = 5861$	$PC = 13472$	$LL = 3$	$TC = 6360$	$PP = 14193$

UL (*LL*) is the upper (lower) bound of the counts of the optimal solution set
TC is the cumulative cost of links that satisfied the budget constraint and *PC* or *PP* is the total performance of these links

check only the number of links $K = 4$ to see whether there exists any combination for which the total profit is more than 14,193. In other words, we now limit the computation need to find the optimal solution to C(20, 4) iterations.

Next, we reduce the candidate set via the concept of variable dominance. If a link is dominated by other links four or more times (the number of links in the optimal solution set), then we can drop this link, which will definitely not appear in the optimal solution set. On the other hand, if a link is superior to other links $20 - 4 = 16$ or more times, then we must retain this link in the optimal solution set. Using the reduced candidate set search, we rule out nine links $\{2, 5, 9, 10, 11, 12, 14, 15, 18\}$ (see *Table 2*) and the remaining candidate links are $\{1, 3, 4, 6, 7, 8, 13, 16, 17, 19, 20\}$, i.e. we further tighten the computation from C(20,4) to C(11,4) iterations without losing optimality. Furthermore, when solving the optimal solution from the reduced candidate set, we can add the remaining dominance inequalities to the constraints (see *Table 3*). In this example, we can add 12 new constraints, such as $x_3 - x_4 \geqslant 0$, $x_3 - x_{20} \geqslant 0$, $\ldots, x_{17} - x_{16} \geqslant 0$, into the optimality search algorithm, which will make the search procedure more efficient.

Note that from the remaining candidate links, the decision rule to keep a link which is definitely in the optimal solution set satisfies $WIN(i) \geqslant 11 - 4 = 7$, and the decision rule to rule out a link which is not in optimal solution set is still the same ($LOSE(i) \geqslant 4$). Although none of any remaining link can satisfy these decision rules (see *Table 3*), but it seems to be true that the optimal solution set has the highest probability to include links $\{7, 8\}$ and not include links $\{4, 16, 19, 20\}$. Why? This is because the count of these links dominate (or is dominated by) all other links closer to the decision rule, i.e. these links have the highest priority to keep (or discard) in the optimal solution set. By using this reduction procedure, we can further tighten the computation from C(11,4) to C(5,2) iterations. Of course, in this step it would be taking a bit of a chance of losing optimality, since it is not easy to define exactly how 'close' the count of each link is to the decision rule. But it can be justified that this method has the highest probability of reaching optimality, and the computation time is reduced significantly. In the case of $N = 20$, for example, the exact optimal solution set is $\{3, 6, 7, 8\}$; this set is included in our reduced candidate set and only needs $C(5, 2) = 10$ iterations to search for optimality.

When the resources are sufficient, such as $N = 23$ and $B = 11500$ in Yang's paper, for example, we tighten the number of links to $11 \leqslant K \leqslant 12$. The maximum total profit is 35,059 when $K = 11$ (see *Table 4*), so we need to check only the number of links $K = 12$ to see

Table 2 Matrix of variable dominance ($N = 20$, $B = 7000$)

Link	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*WIN*(*i*)
1	0	0	0	0	1	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	6
2	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2
3	0	1	0	1	1	0	0	0	1	1	1	1	0	1	1	0	0	1	0	1	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	1	0	1	1	0	0	0	1	1	1	1	0	1	1	0	0	1	1	1	2
8	1	1	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	4
9	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	2
11	0	0	0	0	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	4
12	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	3
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
17	0	0	0	0	1	0	0	0	0	1	0	1	0	1	1	1	0	0	0	0	6
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	1	0	0	1	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	7
20	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	4
LOSS(*i*)	1	4	0	2	10	0	0	0	4	6	5	7	1	16	9	3	1	5	2	2	

Table 3 Matrix of remaining dominance variable ($N = 20$, $B = 7000$)

Link	*1*	*3*	*4*	*6*	*7*	*8*	*13*	*16*	*17*	*19*	*20*	*WIN*(*i*)
1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	1	2
4	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0	1	1	3
8	1	0	0	0	0	0	1	1	1	1	0	5
13	0	0	0	0	0	0	0	1	0	0	0	1
16	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	1	0	0	0	1
19	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0
LOSS(*i*)	1	0	2	0	0	0	1	3	1	2	2	12

Table 4 Data set for $N = 23$, $B = 11500$

Ascending sorting by cost			*Descending sorting by performance*		
Link	*Cost*	*Performance*	*Link*	*Cost*	*Performance*
7	693	3314	7	693	3314
14	787	2746	15	1031	3304
1	796	3191	18	1187	3295
23	801	2762	12	916	3271
8	823	3074	1	796	3191
4	847	3104	11	891	3140
11	891	3140	13	957	3132
12	916	3271	17	1082	3127
13	957	3132	9	1498	3107
20	964	2812	4	847	3104
5	1003	2947	8	823	3074
21	1012	2868	2	1225	2963
15	1031	3304	5	1003	2947
6	1057	2902	3	1066	2916
3	1066	2916	6	1057	2902
16	1067	2870	19	1107	2900
17	1082	3127	16	1067	2870
19	1107	2900	21	1012	2868
18	1187	3295	10	1242	2824
22	1196	2818	22	1196	2818
2	1225	2963	20	964	2812
10	1242	2824	23	801	2762
9	1498	3107	14	787	2746
UL = 12	*TC* = 10490	*PC* = 36361	*LL* = 11	*TC* = 10721	*PP* = 35059

whether there exists any combination with total profit of more than 35,059. That is, we now limit the computation need to find the optimal solution to C(23,12) iterations.

Next, we reduce the candidate set via the concept of variable dominance. Since links {10, 19, 22} are dominated by other links 12 or more times (the number of links in optimality), we drop these links. On the other hand, links {1, 7, 11, 12, 13} are superior to the other links $23 - 12 = 11$ or more times, so these links must be retained in the optimal solution set (see *Table 5*). The remaining candidate links are then {2, 3, 4, 5, 6, 8, 9, 14, 15, 16, 17, 18, 20, 21, 23}, i.e. we have further tightened the computation from C(23,12) to C(15,7) iterations without losing optimality. Furthermore, we can add 31 dominance inequalities (see *Table 6*), such as $x_3 - x_{16} \geqslant 0$, $x_4 - x_2 \geqslant 0, \ldots, x_{18} - x_9 \geqslant 0$, into the optimality search algorithm, making the search procedure more efficient.

Note again that from the remaining candidate links, the decision rule will become $WIN(i) \geqslant 15 - 7 = 8$ and $LOSE(i) \geqslant 7$, respectively. If we take a slight risk of losing optimality, it seems to be a good decision to include links {4, 5, 8, 15} and not {2, 3, 6, 16}, since these links are the best (or worst) links which dominate (or are dominated by) other links many times. In the case of $N = 23$, for example, we use this reduction procedure and tighten the computation from C(15,7) to C(7,3) iterations. Unfortunately, the exact optimal solution set is {1, 2, 4, 5, 7, 8, 11, 12, 13, 15, 17, 18}; this set differs from our reduced candidate set only by link {2}. But this result was just the same as that in Yang's[1] paper, and our method only takes C(7,3) = 35 iterations rather than C(20,12), as does Yang's.

In *Table 7*, we compare the reduced candidate set obtained for the four examples in Yang's paper using the original and revised CRCS (one can apply the same method with $N = 15$ and $N = 28$). *Table 7* clearly shows that the revised CRCS method is superior to Yang's method. When the available resources or budget is insufficient, for example ($N = 15$, 20, 28), the number of links in the reduced candidate set is almost the same for the two methods, and our method not only does not miss the optimal solution, but also generates dominance inequalities to improve the search for optimality. On the other hand, when the available budget is large enough ($N = 23$), the reduced candidate set obtained using the revised CRCS method is smaller than that produced by Yang's method, and it retains all the information on optimality. Furthermore, in this case we have 31 dominance inequalities in the remaining links, which will certainly improve the efficiency of the search for optimality.

CONCLUDING REMARKS

In this paper, we have revised the CRCS algorithm proposed by Yang[1] using information on the cost and performance of all links to tighten the number of links, and by using the concept of variable dominance to reduce the number of candidate links.

From a comparison of the results of Yang's algorithm and the revised

Table 5 Matrix of variable dominance ($N = 23$, $B = 11500$)

Link	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*WIN(i)*
1	0	1	1	1	1	1	0	1	1	1	1	0	1	0	0	1	1	0	1	1	1	1	1	17
2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0	4
4	0	1	1	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	1	1	0	10
5	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	1	1	0	7
6	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0	4
7	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	22
8	0	1	1	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	1	1	0	10
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	1	1	0	1	1	0	0	1	1	0	0	1	0	0	1	1	0	1	1	1	1	0	13
12	0	1	1	0	1	1	0	0	1	1	0	0	1	0	0	1	1	0	1	1	1	1	0	13
13	0	1	1	0	1	1	0	0	1	1	0	0	0	0	0	1	1	0	1	1	1	1	0	12
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	1	1	0	0	1	0	0	1	1	0	0	0	0	0	1	1	1	1	0	0	1	0	10
16	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	2
17	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	5
18	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	4
19	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	2
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	2
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOSS(i)	1	10	9	2	7	9	0	2	8	17	2	1	4	1	1	11	6	2	12	7	8	16	2	

Table 6 Matrix of remaining dominance variable ($N = 23$, $B = 11500$)

Link	*2*	*3*	*4*	*5*	*6*	*8*	*9*	*14*	*15*	*16*	*17*	*18*	*20*	*21*	*23*	*WIN(i)*
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
4	1	1	0	1	1	0	0	0	0	1	0	0	1	1	0	7
5	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	4
6	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
8	1	1	0	1	1	0	0	0	0	1	0	0	1	1	0	7
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	1	1	0	0	1	0	1	0	0	1	1	1	0	0	0	7
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2
18	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOSS(i)	5	4	0	2	4	0	3	0	0	6	1	1	2	3	0	31

Table 7 Comparison of CRCS and *CRCS′* methods

N	*Algorithm*	*Reduced candidate set*	*New constraints*
15	CRCS	1, 2, 8, 14, 15	—
	CRCS′	1, 2, 8, 12, 14, 15	1
	Optimality	8, 12	
20	CRCS	1, 3, 4, 6, 7, 8, 13, 17, 19, 20	—
	CRCS′	1, 3, 4, 6, 7, 8, 13, 16, 17, 19, 20	12
	Optimality	3, 6, 7, 8	
23	CRCS	1, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23	—
	CRCS′	2, 3, 4, 5, 6, 8, 9, 14, 15, 16, 17, 18, 20, 21, 23	31
	Optimality	1, 2, 4, 5, 7, 8, 11, 12, 13, 15, 17, 18	
28	CRCS	3, 5, 10, 12, 13, 17, 25, 26, 27	—
	CRCS′	3, 5, 10, 12, 13, 15, 16, 17, 25, 26, 27	4
	Optimality	15, 16, 27	

CRCS algorithm, we conclude the following.

1. The revised constrained range search method obtains the lower and upper bounds of optimality more accurately and tightly than Yang's method, making further squeezing procedures unnecessary.
2. The count of the reduced candidate set is almost the same in Yang's CRCS and the revised CRCS, but our method ensures that the optimal solution is in the reduced set.
3. When we use the reduced candidate set to find the optimal solution, we can use information on the dominated sets to generate new constraints in order to make the search procedure more efficient, i.e. if we know link i dominates link j, then we can generate the new constraint $x_i - x_j \geqslant 0$, and thereby make the optimality search procedure more efficient.

Note that with both Yang's original CRCS search method and our revised method, the tightness of the number of links is the key factor that determines which links should be retained or discarded in the reduced candidate set search. Under what circumstances will the lower bound be far away from the upper bound in the constrained range search? The answer will depend upon the variation in the link cost: if the cost of the links varies from 0 to the total budget (the worst case), then the constrained range search method will not find the exact number of links easily, and the reduced candidate search method will become less effective (although the decision rule to rule out links which satisfy $LOSE(i) \geqslant UL$ will still be valid). In this case, however, it would be equally as difficult to solve this problem using other methods such as dynamic programming or the Lagrangian relaxation method.

REFERENCES

1 Yang, C 'Link enhancement using constrained range and reduced candidate set searches', *Comput. Commun.*, Vol 15 No 9 (November 1992) pp 573–580

2 Dudzinski, K and Walukiewicz, S 'Exact methods for the knapsack problem and its generalizations', *Euro. J. Operat. Res.*, Vol 28 (1987) pp 3–21

3 Balas, E and Zemel, E 'An algorithm for large zero-one knapsack problems', *Operat. Res.*, Vol 28 (1980) pp 1130–1154