Algorithmic complexity of two defence budget problems

R. Taylor [a]

[a] *Defence Science and Technology Organisation (DSTO), ACT 2600, Australia*
Email: richard.taylor@dsto.defence.gov.au

Abstract: A fundamental challenge in the development of defence capability is to decide on a collection of projects that represent the best value within a given budget constraint. A complicating factor in doing this is taking account of the inter-relationships between projects when assessing value. We investigate two such models. The first assigns value based on subsets of projects that come together to provide effects, the second assigns value through the intermediary of scenarios. In terms of recognized combinatorial optimization problems the first is a form of the Set Union Knapsack problem while the second appears to be a new problem we call the Budget Scenario problem. We analyse the known results about the algorithmic complexity of these problems by showing their relationships with existing problems and known approximation and inapproximability results. We also provide new approximation results for both problems. The main results of this paper are summarised as follows.

When prospective projects mature into operational capabilities they typically come together in subsets to provide joint effects. Thus capabilities that work together and depend upon eachother provide extra value than that provided by each capability in isolation. It seems natural then to balance the cost of a collection of projects against the value provided by the subsets in the collection. This problem is a form of the Set Union Knapsack Problem which generalises a number of well known combinatorial problems in two broad classes. The Knapsack, Subset-Sum and Partition problems have a *polynomial time approximation scheme PTAS* (see Garey and Johnson (1979) and Vazirani (2003)). That is for any given $\epsilon > 0$ there is an algorithms that approximates Knapsack to within $1 - \epsilon$ of the optimal with a run time that bounded by a polynomial of the input size. Further it has a *fully polynomial time approximation scheme FPTAS* with an approximation algorithm with run time bounded by a polynomial of both the input size and $1/\epsilon$. On the other hand Quadratic Knapsack, Weighted Clique, maximum Edge Weighted Clique, and Densest k-Subgraph does not admit a PTAS assuming that random 3-SAT formulas are hard to refute Feige (2002), or if NP does not have randomized algorithms that run in sub-exponential time Knot (2004). On the positive side there is an approximation algorithms for DkS that is within the ratio $O(n^{1/4+\epsilon})$ of the optimal and runs in time $n^{O(1/\epsilon)}$ Bhaskara et al. (2010). We show that the The Set Union Knapsack Problem with subsets of size at most m has an approximation algorithm with run time of at most $O(\frac{n^{m+2}}{\epsilon})$ with an approximation ratio of at least $\frac{m!(1-\epsilon)}{n^{m-1}}$.

In the Budget Scenario problem a list of initiatives is provided each with an anticipated cost. Each initiative is scored against a number of scenarios with a value indicating how useful the initiative is against that scenario. For a collection of initiatives the total value is calculated by summing the best value obtained by any initiative for that scenario in the collection. This initiative can be thought of as the best tool in the toolbox (collection) for the particular job (scenario), while the total value reflects the ability of the toolbox (initiative collection) to address any single job (scenario). The Budget Scenario problem is shown to generalise the established problems of Budgeted Maximum Coverage, Maximum Coverage, Weighted Set Cover and Set Cover. In Khuller, Moss and Naor (1999) an approximation algorithm for Budgeted Maximum Coverage is presented and shown to provide an approximation factor of $(1 - \frac{1}{e})$ of the optimal solution. On the other hand all of the problems in Figure 2 are NP-hard. Also Khuller, Moss and Naor (1999) and Feige (1996) show that if Maximum Coverage, respectively Set Cover is approximable within a factor of $(1 - \epsilon) \log n$ for any $\epsilon > 0$ then $NP \subseteq DTIME(n^{loglogn})$. It follows that the same must be true of Budget Scenario, Budgeted Maximum Coverage, and Weighted Set Cover. We prove that for any given $\epsilon > 0$ the Budget Scenario Problem $G(X, Y)$ has a polynomial approximation algorithm that achieves a factor of $(1 - \frac{1}{e} - \epsilon)$ of the optimal solution.

Keywords: *Set union knapsack, budgeted maximum coverage, approximation, inapproximation*

1 INTRODUCTION

A fundamental challenge in the development of defence capability is to decide on a collection of projects that represent the best value within a given budget constraint. A complicating factor in doing this is taking account of the inter-relationships between projects when assessing value. We investigate two such models. The first assigns value based on subsets of projects that come together to provide effects, the second assigns value through the intermediary of scenarios. In terms of recognized combinatorial optimization problems the first is a form of the Set Union Knapsack problem while the second appears to be a new problem we call the Budget Scenario problem. We analyse the known results about the algorithmic complexity of these problems by showing their relationships with existing problems and known approximation and inapproximability results. We also provide new approximation results for both problems.

2 THE SET UNION KNAPSACK PROBLEM

When prospective projects mature into operational capabilities they typically come together in subsets to provide joint effects. Thus capabilities that work together and depend upon eachother provide extra value than that provided by each capability in isolation. It seems natural then to balance the cost of a collection of projects against the value provided by the subsets in the collection. This problem is modeled as a form of the Set Union Knapsack Problem which is a generalization of a number of well-established optimization and decision problems. These include Maximum Weighted k-Clique, Weighted Dense k-Subgraph, Densest k-Subgraph, Maximum Clique, Knapsack, Subset Sum and Partition. We discuss what is known about the algorithmic complexity of this problem including approximation and inapproximability results. Each problem is stated as input data and a corresponding question about that data.

Set Union Knapsack-(SUK)
Instance: A set S of n items with a cost $c(i)$ associated with each item i; a collection of subsets P of S each with a value $v(j)$ associated with each $j \in P$; a cost bound B. B as well as all costs and values are non negative reals.
Optimization: Find a subset U of S whose total cost is bounded by B with total value maximized. In mathematical terms this can be stated as

$$\max_{U \subseteq S} \left\{ \sum_{j \in P, j \subseteq U} v(j) \right\} \tag{1}$$

$$\text{subject to } \sum_{i \in U} c(i) \leq B. \tag{2}$$

We may consider this system as a weighted hypergraph on the vertex set S with cost weights on the vertices, and value weights on the hyper-edges (vertex subsets P). Set Union Knapsack (SUK) is a natural generalization of the Quadratic Knapsack Problem (QKP) which in turn generalizes a number of other well known problems. In Figure 1 we illustrate these relationships with a link indicating the higher problem as a generalization of the linked problem below it. We state each problem briefly and show restrictions to SUK that show these inclusions (the nested structure of the restrictions corresponding to links). In some cases a problem can be solved by restrictions of another problem together with the need to apply the second problem a small number of times (polynomial in the input length). This is indicated with an asterix next to the first problem in the figure. The precise form of the restrictions that underly Figure 1 are as follows:

Densest k-Subhypergraph-(DkSH) Given a hypergraph and an integer k, determine the set of k nodes such that the subhypergraph induced by this set has a maximum number of hyper-edges.
Restriction: Set all item costs to 1, and all edge values to 1. Set B=k.
Quadratic Knapsack-(QK) A weighted graph is given with costs associated with the vertices and weights associated with both vertices and edges. Given a bound B find a collection of vertices with total cost at most B with maximum total weight in the induced subgraph.
Restriction: Restrict P to 1 and 2 element subsets.
Knapsack Given a collection of items with both associated costs and weights. Given a bound B find a subset of items with total cost at most B that maximizes the total weight.
Restriction: Restrict P to 1 element subsets.
Subset-Sum-(SS) Given a collection of items with associated weights and a number B find a subset of items

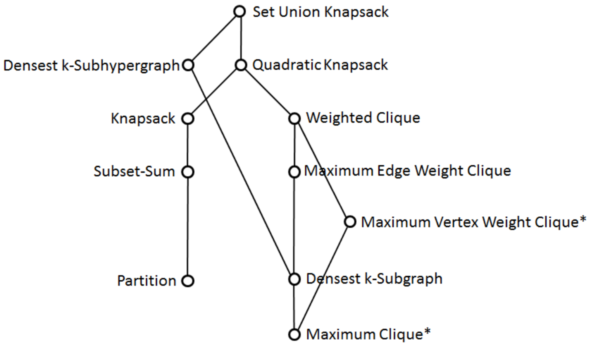

Figure 1. Set Union Knapsack and related problems - asterix indicating decision problem

with total weight equal to B.
Restriction: Restrict P to 1 element subsets. Set both the costs and values of the items equal to the weights. Determine if the optimized value returned by SUK is equal to B.
Partition
Given a collection of items with associated weights find a subset of items with total weight equal to half the sum of all weights.
Restriction: Restrict P to 1 element subsets. Set both the costs and values of the items equal to the weights. Set B to be one half the sum of all costs. Determine if the optimized value returned by SUK is equal to B.
Weighted Clique-(WC) Given a complete graph with both weighted vertices and edges and a bound B find a subgraph with at most B vertices for which the sum of vertex and edge weights is a maximum.
Restriction: Restrict P to 1 and 2 element subsets. Set all item costs to 1.
Maximum Edge Weight Clique-(MEWCP) Given a complete graph with weighted edges and a bound B find a subgraph with at most B vertices for which the sum of edge weights is a maximum.
Restriction: Restrict P to 2 element subsets. Set all item costs to 1.
Maximum Vertex Weight Clique-(MVCP) Given a graph with vertex weightings find a clique with maximum total vertex weight.
Restriction: Restrict P to 1 and 2 element subsets. Let L be twice the sum of all vertex weights. Weight each edge L. Now find the largest B for which the optimized value returned by SUK is of the form $\binom{B}{2}L + C$ for $C \geq 0$ (ensuring that this must be a clique). This can be done with binary search method by solving SUK at most log(n) times. The maximum vertex weight clique for the original problem is then C.
Densest k-Subgraph-(DkS) Given a graph with and a bound k find a subgraph with at most k vertices for which the number of edges is a maximum.
Restriction: Restrict P to 2 element subsets. Set all item costs to 1, and all edge values to 1. Set B=k.
Maximum Clique-(MC) Find the largest clique in a graph.
Restriction: Restrict P to 2 element subsets. Set all item costs to 1, and all edge values to 1. Find the largest B for which the optimized value returned by SUK is $\binom{B}{2}$. This can be done with binary search method by solving SUK at most log(n) times. B is the maximum clique size.

We note that while all of these problems are NP-hard they differ in terms of their known approximation algorithms. Thus in Figure 1 the right hand branch of the problems below the QK seem much harder to approximate than those of the left hand branch. In particular the Knapsack problem has a *polynomial time approximation scheme PTAS* (see Garey and Johnson (1979) and Vazirani (2003)). That is for any given $\epsilon > 0$ there is an algorithm that approximates Knapsack to within $1-\epsilon$ of the optimal with a run time that is bounded by a polynomial of the input size. Further it has a *fully polynomial time approximation scheme FPTAS* with an approximation algorithm with run time bounded by a polynomial of both the input size and $1/\epsilon$. On the other hand Densest k-Subgraph DkS (and so all of its generalizations) does not admit a PTAS assuming that random 3-SAT formulas are hard to refute Feige (2002), or if NP does not have randomized algorithms that run in sub-exponential time Knot (2004). On the positive side there is an approximation algorithms for DkS that is within the ratio $O(n^{1/4+\epsilon})$ of the optimal and runs in time $n^{O(1/\epsilon)}$ Bhaskara et al. (2010).

Table 1. Initiative Scenario Table

initiative	Cost	Scenario 1 Pr 0.3	Scenario 2 Pr 0.1	Scenario 3 Pr 0.4	Scenario 4 Pr 0.2
O	1	10	1	8	2
M	1	7	1	3	4
N	3	6	3	5	6
D	4	8	7	1	2
J	7	4	3	7	9
I	6	8	7	4	1
B	8	8	8	3	10
L	11	10	5	6	9
E	13	8	4	9	7
A	12	7	2	9	1
H	14	1	5	9	9
G	17	9	5	8	4
C	14	5	7	5	7
K	20	6	4	9	4
F	16	1	7	9	1

2.1 An approximation algorithm

An approximation algorithm for SUK can be found by a simple adaption of the dynamic programming approach successful in the Knapsack problem (see Ibara and Kim (1975), and the more recent presentation in a volume dedicated to approximation algorithms Vazirani (2003)). This is parameterized by the maximum number of items in any element of P. Thus for example QKP has at most 2 items in every item subset of P. The proof of the following is omitted due to space limitations and will appear elsewhere.

Theorem 1 The Set Union Knapsack Problem with subsets of size at most m has an approximation algorithm with run time of at most $O(\frac{n^{m+2}}{\epsilon})$ with an approximation ratio of at least $\frac{m!(1-\epsilon)}{n^{m-1}}$.

For QK for example we have an approximation ratio of at least $\frac{2(1-\epsilon)}{n}$ and a run time of at most $O(\frac{n^4}{\epsilon})$. We note that this does not compare well with the approximation ratio of $O(n^{1/4+\epsilon})$ obtained for DkS, but we have not been able to adapt the method used there to take account of the vertex costings that are part of QK (and SUK). We also note that this is the only approximation algorithm for QK or SUK that the author is aware of.

3 THE BUDGET SCENARIO PROBLEM

In two articles Order (2007), Order (2009), a problem is presented and analysed through a series of fictional discussions between clients and analysts. What appears at first to be a prioritization problem is eventually seen to be a budget-value problem in which the best value for a range of budgets is sought. A list of initiatives is provided each with an anticipated cost. Each initiative is scored against a number of scenarios with a value of 1-10 indicating how useful the initiative is against that scenario. As well each scenario is assigned a probability indicating the likelihood that that scenario is occurring at any given time. Table 1 is reproduced from Order (2009). For a collection of initiatives the total value is calculated by summing the product of each scenario probability by the best value obtained by any initiative for that scenario in the collection. This initiative can be thought of as the best tool in the toolbox (collection) for the particular job (scenario), while the total value reflects the expected ability of the toolbox (initiative collection) to address any single job (scenario). Thus the initiatives O, M, and N have a total cost of $1 + 1 + 3 = 5$ and a total value of $0.3 * 10 + 0.1 * 3 + 0.4 * 8 + 0.2 * 6 = 7.7$. We shall explore the mathematics of this optimization problem in terms of what is known about this and related problems, in terms of the algorithmic complexity of their solution. First we observe that the scenario probabilities can be effectively absorbed into the values by multiplying the two together. This is shown in Table 2. Then the total value is calculated more simply by summing the best value obtained by any initiative in the collection. This problem can be modeled as a weighted bipartite graph $G(X, Y)$ with initiatives X and scenarios Y in which there are weights c_i (costs)

Table 2. Initiative Scenario Table

initiative	Cost	Scenario 1	Scenario 2	Scenario 3	Scenario 4
O	1	3.0	0.1	3.2	0.4
M	1	2.1	0.1	1.2	0.8
N	3	1.8	0.3	2.0	1.2
D	4	2.4	0.7	0.4	0.4
J	7	1.2	0.3	2.8	1.8
I	6	2.4	0.7	1.6	0.2
B	8	2.4	0.8	1.2	2.0
L	11	3.0	0.5	2.4	1.8
E	13	2.4	0.4	3.6	1.4
A	12	2.1	0.2	3.6	0.2
H	14	0.3	0.5	3.6	1.8
G	17	2.7	0.5	3.2	0.8
C	14	1.5	0.7	2.0	1.4
K	20	1.8	0.4	3.6	0.8
F	16	0.3	0.7	3.6	0.2

associated with each $i \in X$, and weights w_{ij} (values) associated with each edge $ij, i \in X, j \in Y$. Given a bound (budget) B the Budget Scenario BS Problem can be stated as

Budget Scenario-(BS)
Instance: A set X of items with a weight (cost) c_i associated with each item i; a collection of scenarios Y; weights w_{ij} (values) associated with each edge $ij, i \in X, j \in Y$; a cost bound B. B as well as all costs and weights are non negative reals.
Optimization: Find a subset S of X whose total cost is bounded by B with total value maximized. In mathematical terms

$$\max_{S \subseteq X} \left\{ \sum_{j \in Y} \max_{i \in S} \{w_{ij}\} \right\} \tag{3}$$

$$\text{subject to } \sum_{i \in S} c_i \leq B. \tag{4}$$

This problem is a generalization of a number of well-established optimization and decision problems. These include Budgeted Maximum Coverage which in turn generalizes Weighted Set Cover, Maximum Coverage, and Set Cover. We discuss what is known about the algorithmic complexity of these problems including approximation and inapproximability results. As before in Figure 2 we illustrate these relationships with a link indicating the higher problem as a generalization of the linked problem below it. The statement of each problem is given and restrictions to BS that show these inclusions are summarized.

Budgeted Maximum Coverage-(BMC) Given a collection S of sets with associated costs defined over a domain of weighted elements, and a budget B, find a subset $S' \subseteq S$ such that the total cost of sets in S' does not exceed B and the total weight of elements covered by S' is maximized.
Restriction: Restrict w_{ij} to 0 or y_j for all i, j for some positive weights y_j. In effect this is the same as the elements of Y each having a weight.
Weighted Set Cover-(WSC) Given a collection S of sets with associated costs defined over a domain of elements, find a subset $S' \subseteq S$ such that the total cost of sets in S' is minimized and that covers all elements in the domain.
Restriction: Restrict w_{ij} to 0 or 1 for all i, j. Let the sum of the costs c_i in X be C. We wish to find the smallest B for which the BS gives a solution of $|Y|$. This can be done with a binary search method by solving BS at most log(C) times.
Maximum Coverage-(MC) Given a collection S of sets defined over a domain of elements and an integer B, find a subset $S' \subseteq S$ of at most B subsets that covers a maximum number of elements in the domain.
Restriction: Set $w_{ij} = 0$ or 1 for all i, j. Set all costs to 1.

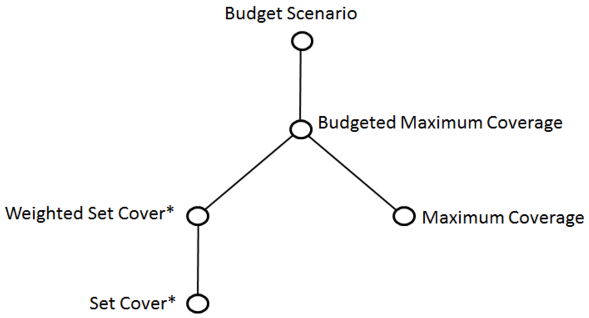

Figure 2. Budget Scenario and related problems - asterix indicating decision problem

Set Cover-(SC) Given a collection S of sets defined over a domain of elements, find a subset $S' \subseteq S$ of as few subsets as possible that covers the domain of elements.
Restriction: Set $w_{ij} = 0$ or 1 for all i, j. Set all costs to 1. We wish to find the smallest B for which the BS gives a solution of $|Y|$. This can be done with a binary search method by solving BS at most log($|X|$) times.

In Khuller, Moss and Naor (1999) an approximation algorithm for BMC is presented and shown to provide an approximation factor of $(1 - \frac{1}{e})$ of the optimal solution. On the other hand all of the problems in Figure 2 are NP-hard. Also Khuller, Moss and Naor (1999) and Feige (1996) show that if MC, respectively SC is approximable within a factor of $(1 - \epsilon)$ log n for any $\epsilon > 0$ then $NP \subseteq DTIME(n^{loglogn})$. It follows that the same must be true of BS, BMC, and WSC.

3.1 An approximation algorithm

We note that BS can be solved exactly provided the number of initiatives $|X|$ or the number of scenarios $|Y|$ are small enough. Specifically the optimal solution can be found in time $O(2^{|X|}|Y|)$, and in time $O([\binom{|X|}{|Y|} + \binom{|X|}{|Y|-1} + .. + \binom{|X|}{1}]|Y|)$. The first expression follows from simply trying each subset of X initiatives. For the second we observe that any selection of more than $|Y|$ initiatives must contain at least one initiative that is not the best value for any scenario. This initiative can therefore be removed from the subset without reducing the total value. Thus it is sufficient to consider sets of $|Y|$ or less initiatives. For our application we might expect the number of scenarios to be small in comparison to the number of initatives. If this is the case the second expression is much smaller than the first and is approximately $O(\frac{|X|^{|Y|}}{(|Y|-1)!})$.

In the following we show how BS problems can be transformed into instances of BMC and so take advantage of the approximation algorithm of Khuller, Moss and Naor (1999). Let $G(X, Y)$ be an instance of BS with $n = |X|$ and $m = |Y|$. We shall also assume that each edge weight w_{ij} is a non-negative integer (if not simply multiply all edge weights by a suitable power of 10). Also $w_{ij} = 0$ indicating no edge between vertices i and j. For each $j \in Y$ assume without loss of generality that $w_{1j} \leq w_{2j} .. \leq w_{nj}$. Form a new graph G' by replacing each vertex y_j in Y with $r = w_{nj}$ vertices $y'_1, y'_2, .., y'_r$ and making each vertex x_i of X adjacent to vertices $y'_1, y'_2, .., y'_t$ where $t = w_{ij}$. Each edge of this new graph has a unit weight. We now have a new graph $G'(X, Y')$ in which $|Y'| \leq W|Y|$ where W is the largest w_{ij}. It is an easy matter to see that the optimal solutions of G and G' are the same, and that G' is a BMC problem (in fact it is also an MC problem). We note that for our application W is a value indicator and so it seems reasonable that this would be a fixed constant (such as 10 used in Order (2007), Order (2009)). It follows immediately from Khuller, Moss and Naor (1999) that

Theorem 2 For a fixed $W = max\{w_{ij}\}$ the Budget Scenario Problem has a polynomial approximation algorithm that achieves a factor of $(1 - \frac{1}{e})$ of the optimal solution.

However for large W we can also use scaling methods to model a BS problem as a BMC problem with edge weights all 1 in which the optimal solution of the BMC problem is within $1 - \epsilon$ of the corresponding BS problem (details to appear elsewhere). Noting that $(1 - \frac{1}{e})(1 - \epsilon) \geq (1 - \frac{1}{e} - \epsilon)$ we can combine this last result with Theorem 2 to give

Theorem 3 For any given $\epsilon > 0$ the Budget Scenario Problem $G(X, Y)$ has a polynomial approximation algorithm that achieves a factor of $(1 - \frac{1}{e} - \epsilon)$ of the optimal solution.

4 CONCLUSIONS

In this paper we model two defence budget problems as combinatorial optimization problems and discuss their relationship with problems already identified and studied in the literature. The first is a form of the Set Union Knapsack problem while the second appears to be a new problem we call the Budget Scenario problem. Although both of these problems are NP-hard they differ markedly in terms of known approximation and inapproximation results. The Set Union Knapsack with subsets of size at most m has an approximation algorithm with run time of at most $O(\frac{n^{m+2}}{\epsilon})$ with an approximation ratio of at least $\frac{(1-\epsilon)}{n^{m-1}}$. On the other hand since it generalizes Densest k-Subgraph it does not admit a PTAS assuming that random 3-SAT formulas are hard to refute Feige (2002), or if NP does not have randomized algorithms that run in sub-exponential time Knot (2004).

We show the Budget Scenario problem with initiatives X and scenarios Y can be solved exactly in time $O(2^{|X|}|Y|)$, and in time $O(\frac{|X|^{|Y|}}{(|Y|-1)!})$ for $|Y|$ small in comparison to $|X|$. Building on the results of Khuller, Moss and Naor (1999) we show Budget Scenario has a polynomial time approximation algorithm with an approximation ratio of at least $(1 - \frac{1}{e} - \epsilon)$ of the optimal solution. On the other hand since BS generalizes Maximum Cover it follows from Khuller, Moss and Naor (1999) and Feige (1996) that if Budget Scenario is approximable within a factor of $(1 - \epsilon)$ log n for any $\epsilon > 0$ then $NP \subseteq DTIME(n^{loglogn})$.

REFERENCES

Bhaskara, A., Charikar, M., Chlamtac, E. and Vijayaraghavan, A. (2010) Detecting high log-densities - an $o(n^{1/4})$ approximation for densest k-subgraph, *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC'04)* pp. 201–210.

Feige, U. (1996) A threshold of ln(n) for approximating set cover, *Proceedings of the 28th annual ACM Symposium on the Theory of Computing* pp. 314–318.

Feige, U. (2002) Relations between average case complexity and approximation complexity, *Proceedings of the 34th annual ACM Symposium on Theory of Computing (STOC'02)* pp. 534–543.

Garey, M. R. and Johnson, D. S. (1979) *Computers and Intractability - A guide to the theory of NP-Completeness*, W. H. Freeman.

Ibara, O. H. and Kim, C. E. A. (1975) Fast approximation algorithms for the knapsack and sum of subset problems, *J. Assoc. Comput. Mach.* **22**, 463–468.

Khuller, S., Moss, A. and Naor, J. (1999) The budgeted maximum coverage problem, *Information Processing Letters* **70**, 39–45.

Knot, S. (2004) Ruling out ptas for graph min-bisection, densest subgraph and bipartitie clique, *Proceedings of the 44th annual IEEE Symposium on the Foundations of Computer Science (FOCS'04)* pp. 136–145.

Order, N. (2007) The priority list, *PHALANX - The Bulletin of Military Operations Research* **40**(3), 8–10.

Order, N. (2009) The priority list - revisited, *PHALANX - The Bulletin of Military Operations Research* **42**(4), 23–34.

Vazirani, V. (2003) *Approximation Algorithms*, Springer-Verlag Berlin Heidelberg New York.