

# Assessing Supply Chain Robustness Through Stress Testing

Slava Shekh and Luke Marsh

Defence Science and Technology Group, Edinburgh, South Australia  
slava.shekh@dsto.defence.gov.au

## Abstract

Military logistics planning is a complex and time consuming process, and ultimately many factors can impact the devised plan. We have developed a way of assessing the robustness of a logistics plan and associated supply chain by considering the occurrence of disruptive negative events. These events may occur in isolation or simultaneously. Each event is associated with a plan variable and the impact of a negative event can then be assessed by measuring whether the modification of its associated variable causes any stockouts in the plan during simulation. By modifying any two variables simultaneously, we can identify the minimum values of those two variables, which in combination cause a stockout. These minimum values are called breakpoints, and a set of breakpoints is referred to as a breakpoint front. We have investigated various approaches of calculating or estimating breakpoint fronts, including brute force search, the Monte Carlo method, 2D binary search and the multi-objective optimization methods, SPEA2 and NSGA-II. Our experiments showed that 2D binary search performed best overall, due to its low computation time and accuracy in calculating a range of breakpoint fronts.

## Introduction

Military logistics planning is a complex and time consuming process. The formulated plans determine the expected demand in terms of food, fuel, ammunition etc., the supply chain and the transports needed to meet the demand. However, as the old saying goes: “no plan survives contact with the enemy” (Hughes 2009). This is primarily due to the extremely dynamic and volatile environment surrounding military operations. Many variables and factors can change and make the original plan obsolete. For example: What if the proposed logistics routes are denied and alternative routes are required? What if the force size or operational tempo increases and more resources are required to support the force? What if another activity requiring support occurs concurrently? What if nodes in the supply chain experience a reduction in capacity or become unavailable? What if the

prescribed logistics transports become unavailable? What if the stock loading and unloading times of the transports increase? What if the duration of the required logistics support increases? What if there are weather events or other occurrences which delay the delivery of stock? All of these questions are valid and reflect likely occurrences in military operations. Is there a way to cater for these potential occurrences during the planning process?

The aim of this research is to develop a method to analyse the strengths and weaknesses of a given logistics plan against possible disruptive negative events. The information derived from this analysis can then be used during the planning process to create a more robust plan.

## Problem Description

In order to analyse the robustness of a given logistics plan, we need to define a measure of success or failure for a given plan. For this study we will use the measure of “out of stock” events, which are also referred to as stockouts. If any stock (food, fuel, ammunition etc.) runs out at any point in the operation, a stockout occurs and we deem the logistics plan to have failed. To determine whether stockouts will occur in a given logistics plan, we have created a movement simulation. For the purposes of this paper, the simulation can be considered a black box which takes a plan consisting of stock demand, routes, nodes, and transports as inputs, and determines whether that plan is a success or failure. A plan is considered to be a failure if the stock level of any logistics node falls below zero at any time during the simulation.

To represent the negative events that could occur during a military operation, certain plan variables in the simulation can be modified. The plan variables denoted as  $v_1 \dots v_k$  which can be changed include: increase in the lengths of routes; increase in the amounts of daily demand required; increase in the duration of the military operation; reduction in the capacity of logistics nodes to hold stock; reduction in the initial amounts of stock held on transports at the start of the simulation; reduction in the availability of

transports during the simulation; and finally, an increase in the stock loading/unloading time for transports. The authors have determined that changes in these plan variables cover many likely real world events by representing the specific implications they have on a logistics plan. For example, material handling equipment faults, high sea states, and delays in delivery of certain stocks at ports all ultimately affect the loading time of ships, and can be represented by changes to this single plan variable.

In order to find when a stockout occurs for each plan variable, the authors initially implemented a simple brute force search algorithm. Focusing on one plan variable at a time, the brute force search incrementally increases or decreases the plan variable by one percent, runs the movement simulation, and records any stockouts. For example, for the “increased daily demand” plan variable, the daily demand for all stocks is increased in one percent increments. To avoid excessive runtimes of the brute force search algorithm and due to the fact that some plans never stock out, the brute force search is limited to searching up to 200 increments of the plan variables. If a stockout occurs, the value of the plan variable is recorded and the brute force search moves to the next plan variable. The outcome is that brute force search finds the exact ‘breakpoint’ of each variable in the plan. From this analysis, a military planner can determine the strengths and weaknesses of a plan against possible negative events. However, this kind of analysis only considers the occurrence of one negative event at a time, but what if multiple negative events occur simultaneously during an operation?

In the case of two simultaneous negative events, we want to find the value combinations of the two associated plan variables that would cause stockouts in the plan. If the value of one variable is fixed and the other is increased, there will be a point at which the plan stocks out (i.e. the breakpoint). The calculation of all breakpoints for the different values of the variables defines a breakpoint front. The brute force search algorithm, which was used for one plan variable, was extended to calculate this breakpoint front for combinations of plan variables. However, the brute force search was found to be far too inefficient, and therefore other approaches to calculate the breakpoint front were investigated. The brute force search and other approaches are discussed in more detail in the Approaches section.

## Related Work

The relevant literature for this research is in the modelling and simulation of supply chains (Terzi and Cavalieri 2004) and supply chain resilience (Ponis and Koronis 2012; Christopher and Peck, 2004). (Spiegler, Naim, and Wikner 2012) provide a thorough review of related literature and a

detailed exploration of the resilience and robustness of supply chains. (Jain and Leong 2005) stress test a supply chain in a simulation environment with double and quadruple the volume of demand required, in order to show that the supply chain can meet unforeseen spikes in demand. (Schmitt and Singh 2009) analyse disruptions in supply chains using discrete-event simulation, and explore ways to mitigate these disruptions with proactive planning. (Fallasca, Zobel, and Cook 2008) conducted research in developing a quantitative approach for assessing supply chain resilience to disasters. They explored various factors such as the density, complexity and node criticality of supply chains and their effects on the likelihood of disruptions, the impact of disruptions, and the ability of the supply chain to return to normalcy. (Thiagarajan et al. 2011) investigate the impact of losing critical transports on the supply chain via simulation. Although the mentioned literature is relevant to our work, to the best of the author’s knowledge, using breakpoint front calculation to assess supply chain robustness is a novel approach.

## Approaches

The authors have explored a number of approaches for calculating breakpoint fronts, including brute force search, the Monte Carlo method, 2D binary search and multi-objective optimization. In order to calculate a breakpoint front, each approach must have a way of generating combinations of values for two variables. For instance, “increased daily demand” and “increased duration of military operation” are two plan variables. A possible value combination for these variables is a 10% increase in daily demand and a 20% increase in the duration of the operation. Each value combination can be assessed using the movement simulation described in the Problem Description to determine whether that combination causes the plan to stock out. Once a number of combinations have been generated and assessed, the breakpoint front is simply the set of points separating the value combinations that cause stockouts and those that do not.

### Brute Force Search

A simple way of generating combinations of variable values is to simply enumerate all of them using a brute force or exhaustive search. Since this approach covers the entire search space, it will yield an exact solution (assuming a discrete search space). However, looking at all combinations of variable values will have an impact on computational performance, particularly in larger search spaces where variables can take on a wide range of values.

## Monte Carlo Method

The Monte Carlo method is a form of random sampling that can be used to approximate the solution to some problems (Metropolis and Ulam 1949). This can be applied to our breakpoint problem by repeatedly generating random combinations of variable values within the search space. Since this is a random process, the Monte Carlo method will generally yield an approximate solution, the quality of which will depend on the number of random samples. The Monte Carlo method can be much faster than brute force, especially with a small number of samples, but using too few samples can result in poor quality results that do not sufficiently cover the search space.

## 2D Binary Search

Binary search is a logarithmic time algorithm for finding a value in a sorted list by iteratively eliminating half of the remaining data where the value is known not to exist (Knuth 1998). In the context of breakpoint calculation, we can't directly use this method because our problem is two dimensional, and we don't actually know the value that we're looking for (the breakpoint). Therefore, we have developed a variant of binary search, adapted for our particular problem. Our approach assumes that the number of stockouts is monotonically increasing (i.e. an increased variable value will result in an equal or greater number of stockouts). This assumption holds for the plan variables that we have investigated, but may not necessarily hold for other variables found in real world problems. Exploring alternative techniques that don't make this assumption is therefore an area for future work.

Our two-dimensional (2D) binary search works by fixing a variable  $v_1$  (initially to zero), and performing a 1D binary search over the other variable  $v_2$ . Since the specific value that needs to be found is unknown, the algorithm maintains a lower bound value that is known to be stockout-free, and an upper bound value that is known to cause stockout. The algorithm then iteratively calculates the midpoint of these bounds. If the midpoint is stockout-free, the midpoint becomes the lower bound; otherwise, the midpoint becomes the upper bound. This process is repeated until the difference between the upper bound and lower bound is 1, which means the exact breakpoint has been found.  $v_2$  is then fixed to the value of the lower bound, and a 1D binary search, as described above, is performed over  $v_1$ . This process of alternating 1D binary searches is repeated until the entire breakpoint front is found.

This binary search variant was developed to try to overcome the performance limitations of a fully exhaustive search. By focusing the search along the breakpoint front and taking advantage of a binary search strategy, our method examines much fewer combinations of variable values than a brute force search. More specifically, the

time complexity of brute force search for two variables is  $O(n^2)$ , while our 2D binary search is  $O((\log n)^2)$ , resulting in a reduction from quadratic to polylogarithmic time. In addition, since our 2D binary search is based on 1D binary search, it is able to find the exact solution. This assertion is further validated through our experiments.

## Multi-Objective Optimization

In cases where even the 2D binary search is not fast enough, another option is to apply a multi-objective optimization algorithm to the problem. Multi-objective methods are used for problems where multiple objective functions need to be optimized simultaneously. In these kinds of problems, there can be many optimal solutions, known as Pareto optimal solutions, which correspond to different trade-offs of the objective functions. A multi-objective optimization algorithm can be used to estimate the entire set of Pareto optimal solutions, which is also known as the Pareto front.

Our breakpoint problem can be represented as a multi-objective problem by modelling each of our plan variables  $v_1 \dots v_k$  as a decision variable  $d_1 \dots d_k$ , which together form a solution  $x$ . Our goal is to find solutions with variable values that are as high as possible, yet don't encounter stockouts during simulation. These kinds of solutions contain variable values that represent the limits or breakpoints of the variables, and collectively these solutions form the breakpoint front.

Given some solution  $x$ , the solution can be evaluated using our movement simulation to produce a result  $r_x$ , which identifies the number of stockouts that occurred during the simulation. The quality of solution  $x$  can then be evaluated using  $k$  objective functions, which are defined as:

$$f_k(x) = \begin{cases} d_k, & r_x = 0 \\ 0, & \text{otherwise} \end{cases}$$

The goal is to maximize these  $k$  objective functions:

$$\max [f_1(x), \dots, f_k(x)]$$

## Experiments and Results

We have run a number of experiments to assess each of the discussed approaches on their solution quality and computational performance. All experiments were performed on an Intel Core i7-920 Processor (2.66 GHz) with 6GB RAM running Windows 7. All algorithms were implemented in Java.

For the multi-objective optimization, we used a Java-based framework called jMetal (Durillo and Nebro 2011), which implements a range of metaheuristics and multi-objective algorithms. In particular, we experimented with two popular multi-objective algorithms: SPEA2 (Zitzler,

Laumanns, and Thiele 2001) and NSGA-II (Deb et al. 2002). Both are evolutionary algorithms that evolve a population of solutions over many generations to try to find a good approximation of the Pareto front. However, SPEA2 takes the approach of maintaining an external population of non-dominated solutions that is regularly mixed with the active population, while NSGA-II repeatedly sorts the population into fronts based on the non-dominance of solutions.

Our experiments are based on a two-node supply chain consisting of ten different supply items with distinct demand profiles, supported by multiple air and sea transports with varying capabilities. Using this supply chain, we have devised three scenarios that correspond to different combinations of two plan variables. The breakpoint fronts in these scenarios are approximately representative of the kinds of breakpoint fronts that can occur across all scenarios. In particular, the scenarios are:

- An increase in the amount of daily demand, and a surge in demand for some period of time (Scenario 1).
- An increase in the length of routes, and an increase in the duration of the operation (Scenario 2).
- An increase in the amount of daily demand and a reduction in the capacity of logistics nodes to hold stock (Scenario 3).

Since brute force search and 2D binary search produce exact solutions to the breakpoint problem, we wanted to compare these two methods on computational performance. We ran the algorithms over the three scenarios and the results are shown in Table 1.

	Brute force search	2D binary search
<b>Scenario 1</b>	29.611 seconds	0.755 seconds
<b>Scenario 2</b>	49.049 seconds	0.052 seconds
<b>Scenario 3</b>	31.157 seconds	0.903 seconds

Table 1: Performance of brute force vs. 2D binary search.

The results show that 2D binary search is able to compute the breakpoint front for each scenario in under a second, which is significantly faster than a brute force search. The performance exhibited by 2D binary search is fast enough for calculating an individual breakpoint front, but what if we need to calculate a range of breakpoint fronts corresponding to different variable combinations? Perhaps one of the approximation algorithms can provide even better performance at the cost of some solution accuracy.

Based on this premise, we ran a number of experiments to compare 2D binary search with the Monte Carlo method, and the multi-objective optimization algorithms, SPEA2 and NSGA-II. Initially, we compared the break-

point fronts that each algorithm could produce given the same amount of computation time. Using the execution time of 2D binary search as a benchmark, we gave the other algorithms the same amount of time and recorded the breakpoint fronts that they produced. Examples of the kind of breakpoint fronts produced by each method for Scenario 1 are shown in Figure 1. The comparison shows that SPEA2 and NSGA-II are able to produce good approximations for the breakpoint front, while the Monte Carlo solution has a number of gaps due to the limited number of simulations that could be run within the given time window. The results for Scenarios 2 and 3 were comparable.

The results from the previous experiment show that SPEA2 and NSGA-II can produce good approximations for breakpoint fronts in the same amount of time that 2D binary search can produce an exact solution. However, perhaps these multi-objective methods can produce similar quality approximations in a much shorter time. This idea provides the motivation for our next experiment, where we limit the amount of computation time allowed for each approximation algorithm and record the overall breakpoint error. This is achieved by running each algorithm for different numbers of runs (for Monte Carlo this is the number of simulations, and for multi-objective optimization this is the number of generations). The results of this experiment across the three scenarios are shown in Table 2, where each result is the average of 100 runs. The results from brute force are included to validate the accuracy of 2D binary search.

The breakpoint area (BPA) is calculated as the area of the polygon bounded by the breakpoint front (the left part of the graph), which represents the combinations of variable values that don't cause stockouts (see Figure 1 for some examples). The error is calculated as the percentage difference between the breakpoint area of a given algorithm and the breakpoint area produced by an exact method, such as brute force. More generally, breakpoint area can also be used as a measure to compare two given logistics plans, where a larger breakpoint area is indicative of a more robust plan.

The results in Table 2 show that the effectiveness of the algorithms greatly depends on the scenario. In Scenario 2, the 2D binary search is at least twice as fast as the other methods, even when these methods are only run for a small number of evaluations (100). However, in Scenario 1, if a 20-25% error is acceptable, NSGA-II can find a suitable breakpoint front in almost 1/10th of the time compared to 2D binary search. Similarly, the results for Scenario 3 suggest that computation times faster than 2D binary search are possible if a certain amount of breakpoint error is allowed.



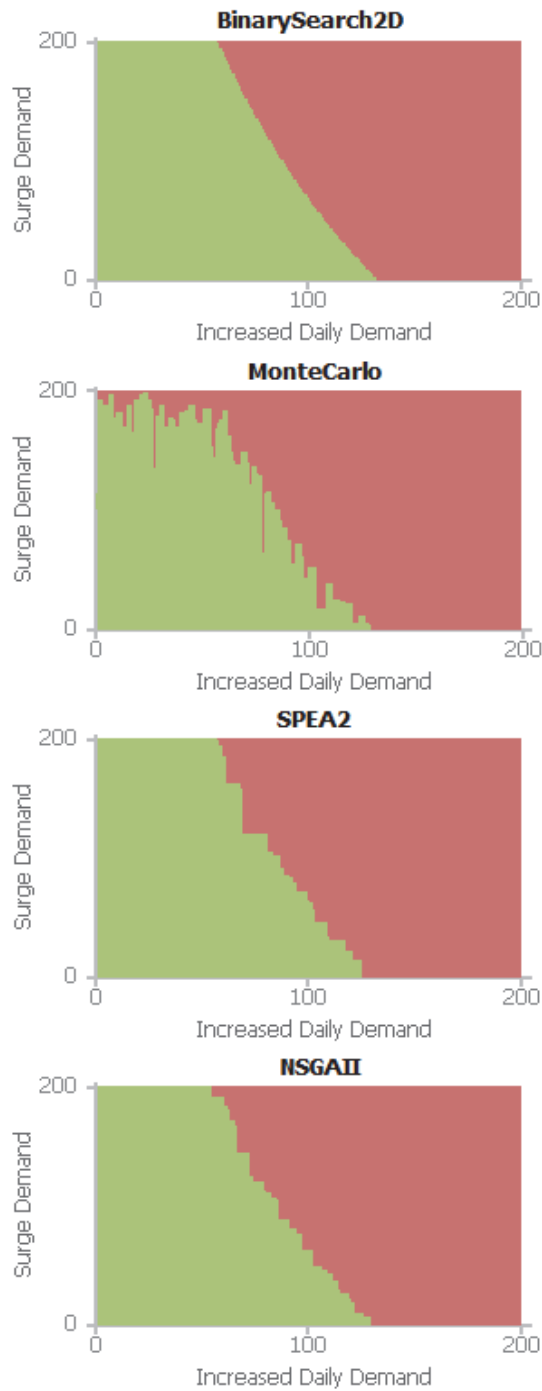


Figure 1: Examples of breakpoint front calculation using different methods (Scenario 1).

Overall, we would recommend 2D binary search for most breakpoint front calculations, due to its accuracy and comparably good performance. Multi-objective optimization may be applicable in situations where dozens or hundreds of breakpoint front calculations are needed very quickly and a relatively high breakpoint error is acceptable.

Scenario 1				
Algorithm	Runs	Comp. Time	BPA	Error
Brute force	-	28.786 seconds	17744	-
2D binary	-	0.676 seconds	17744	0.0%
Monte Carlo	400	0.284 seconds	11144.79	37.2%
	200	0.143 seconds	9922.35	44.1%
	100	0.071 seconds	9166.12	48.3%
SPEA2	400	0.307 seconds	16199.24	8.7%
	200	0.154 seconds	14673.39	17.3%
	100	0.074 seconds	12987.53	26.8%
NSGA-II	400	0.297 seconds	16241.74	8.5%
	200	0.153 seconds	15050.19	15.2%
	100	0.077 seconds	13883.52	21.8%
Scenario 2				
Algorithm	Runs	Comp. Time	BPA	Error
Brute force	-	48.602 seconds	32534	-
2D binary	-	0.051 seconds	32534	0.0%
Monte Carlo	400	0.481 seconds	20848.3	35.9%
	200	0.241 seconds	18387.33	43.5%
	100	0.119 seconds	16974.12	47.8%
SPEA2	400	0.501 seconds	31640.22	2.7%
	200	0.255 seconds	29924.1	8.0%
	100	0.124 seconds	27666.6	15.0%
NSGA-II	400	0.496 seconds	31866.06	2.1%
	200	0.253 seconds	30366.16	6.7%
	100	0.128 seconds	28571.69	12.2%
Scenario 3				
Algorithm	Runs	Comp. Time	BPA	Error
Brute force	-	31.063 seconds	7248	-
2D binary	-	0.889 seconds	7248	0.0%
Monte Carlo	400	0.319 seconds	3986.67	45.0%
	200	0.162 seconds	3723.46	48.6%
	100	0.081 seconds	3383.37	53.3%
SPEA2	400	0.358 seconds	6085.04	16.0%
	200	0.173 seconds	5020.54	30.7%
	100	0.080 seconds	3908.13	46.1%
NSGA-II	400	0.337 seconds	6202.6	14.4%
	200	0.169 seconds	5349.2	26.2%
	100	0.082 seconds	4633.83	36.1%

Table 2: Comparative breakpoint error of different algorithms.

We have also considered the problem of solving  $k$ -variable breakpoint problems for  $k > 2$ . Binary search could be adapted to a larger number of variables, albeit at rapidly increasing algorithmic complexity. However, multi-objective algorithms, such as SPEA2 or NSGA-II, can be adapted quite easily to more variables by simply defining  $k$  decision variables and  $k$  objective functions (see Multi-Objective Optimization section). In the 2-dimensional problem, the error is calculated based on the area of the polygon bounded by the breakpoint front. In  $k$  dimensions, the error could be based on the volume of the polytope bounded by the breakpoint hypersurface. Although this is a non-trivial problem, there are known techniques for computing the volume of convex polytopes (Büeler, Enge, and Fukuda 2000).

## Conclusion and Future Work

We have presented a way of assessing the robustness of a military logistics plan by considering the occurrence of disruptive negative events. These events may occur in isolation or simultaneously. Each event has a corresponding plan variable and the impact of a negative event can be assessed by measuring whether the modification of its associated variable causes any stockouts in the plan. By modifying any two variables simultaneously, we can identify the minimum value of those two variables, which in combination cause a stockout. The set of these combinations defines a breakpoint front. Ultimately, this breakpoint front can assist military planners in determining the exact point where the logistics plan fails. This knowledge can inform the planning process and help to create a more robust logistics plan.

We have investigated various approaches of calculating or estimating breakpoint fronts, including brute force search, the Monte Carlo method, 2D binary search and the multi-objective optimization methods, SPEA2 and NSGA-II. Our experiments showed that the best overall method was 2D binary search, due to its accuracy and low computation times. The multi-objective methods may potentially have even lower computation times, if some breakpoint error is acceptable, and therefore may be applicable in situations where dozens or hundreds of breakpoint front calculations are needed very quickly.

Future work could consider the problem of solving breakpoint problems containing more than two variables, since it's possible for more than two negative events to occur simultaneously. Either a  $k$ -dimensional binary search or one of the multi-objective methods could be adapted for this purpose. The latter would be a simple extension of the approach described in this paper using  $k$  objective functions, though it may be difficult to effectively visualise and present these results to a military planner.

## References

- Büeler, B.; Enge, A.; and Fukuda, K.. 2000. Exact volume computation for polytopes: a practical study. *Polytopes—combinatorics and computation*. 29: 131–154.
- Christopher, M., and Peck, H. 2004. Building the resilient supply chain. *International Journal of Logistics Management* 15(2): 1–14.
- Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. A. M. T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2): 182–197.
- Durillo, J. J., and Nebro, A. J. 2011. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software* 42(10): 760–771.
- Falasca, M.; Zobel, C. W.; and Cook, D. 2008. A decision support framework to assess supply chain resilience. In *Proceedings of the 5th International ISCRAM Conference*, 596–605.
- Hughes, D. ed. 2009. *Moltke on the art of war: Selected writings*. Presidio Press.
- Jain, S., and Leong, S. 2005. Stress testing a supply chain using simulation. In *Proceedings of the 2005 Winter Simulation Conference*, 1650–1657.
- Knuth, D. E. 1998. *The art of computer programming: sorting and searching*. Pearson Education.
- Metropolis, N., and Ulam, S. 1949. The Monte Carlo method. *Journal of the American statistical association* 44(247): 335–341.
- Ponis, S. T., and Koronis, E. 2012. Supply chain resilience: Definition of concept and its formative elements. *Journal of Applied Business Research* 28(5): 921–930.
- Schmitt, A. J., and Singh, M. 2009. Quantifying supply chain disruption risk using Monte Carlo and discrete-event simulation. In *Proceedings of the 2009 Winter Simulation Conference*, 1237–1248.
- Spiegler V. L.; Naim, M. M.; and Wikner, J. 2012. A control engineering approach to the assessment of supply chain resilience. *International Journal of Production Research* 50(21): 6162–6187.
- Terzi, S., and Cavalieri, S. 2004. Simulation in the supply chain context: a survey. *Computers in Industry* 53(1): 3–16.
- Thiagarajan, R.; Kwok, H. W.; Calbert, G.; Gossink, D.; Shekh, S.; and Allard, T. 2011. A simulation-based risk analysis technique to determine critical assets in a logistics plan. In *19th International Congress on Modelling and Simulation*, 503–509.
- Zitzler, E.; Laumanns, M.; and Thiele, L. 2001. SPEA2: Improving the strength Pareto evolutionary algorithm.