Multiobjective Optimization for the Stochastic Physical Search Problem

Jeffrey Hudack[1,2], Nathaniel Gemelli[1], Daniel Brown[1], Steven Loscalzo[1], and Jae C. Oh[2]

[1] Air Force Research Laboratory, Rome, NY, USA
{jeffrey.hudack,nathaniel.gemelli,daniel.brown.81,steven.loscalzo}@us.af.mil
[2] Syracuse University, Syracuse, NY, USA
jcoh@ecs.syr.edu

Abstract. We model an intelligence collection activity as multiobjective optimization on a binary stochastic physical search problem, providing formal definitions of the problem space and nondominated solution sets. We present the Iterative Domination Solver as an approximate method for generating solution sets that can be used by a human decision maker to meet the goals of a mission. We show that our approximate algorithm performs well across a range of uncertainty parameters, with orders of magnitude less execution time than existing solutions on randomly generated instances.

Keywords: path planning, planning under uncertainty, multiobjective optimization, stochastic search

1 Introduction

We address the challenge of utilizing Intelligence, Surveillance, and Reconnaissance (ISR) assets to locate and identify Anti-satellite weapons (ASAT), which we refer to as AS-ISR. Sites that support ASAT require fixed infrastructure to deploy, but can house mobile equipment that may be located at any site. Construction of the proper infrastructure only indicates that the site is capable of housing the weapon, but provides no guarantee that it will be located there. Due to the high cost of the equipment, and a motivation to not have a small easily identified set of target, there are often more sites than platforms. Therefore, there is uncertainty surrounding the existence of an ASAT that can only be dispelled by sending a sensor-laden platform to the site. This problem can be represented as a planar graph, with vertices representing the site of interest and edges denoting the travel cost between sites.

The AS-ISR problem seeks solutions that minimize cost and minimize the probability of failure, objectives that are in conflict. Additionally, conducting these missions requires human oversight to assure objectives are met. The decision maker may have a preference prior to search, allowing the search to focus on a single objective. But commonly, the decision maker may wish to evaluate a

set of alternatives after analyzing a set of alternatives against their own preference [9]. In the latter case, a *solution set* needs to be generated that spans the objective space.

If the decision maker is able to provide additional budget or probability of failure constraints, the goal is to find a path that meets one of two objective functions, both of which have been shown to be NP-complete [8] on general graphs:

- **Minimize budget**: given a required probability of failure p^*_{fail}, minimize the budget necessary to ensure an active site is located with certainty of at least $1 - p^*_{fail}$. This answers the question, "How much budget will I need to ensure the risk of failure is below acceptable levels?"
- **Minimize probability of failure**: given a fixed starting budget B^*, minimize the probability of failing to find an active site, while ensuring the sum of travel and purchase costs do not exceed B^*. This addresses, "What risk of failure can I expect given this limited budget?"

A user may want to be presented with a range of options representing the trade-off between budget and risk. For example, the mission may be a part of a larger operation and is one of many competing objectives, which requires evaluating the overall efficiency of the search process to determine if it should be carried out at all, or if additional resources are needed. If a decision maker has a requirement for a maximum travel cost, or minimum risk, we can optimize the solution for the other objective. Otherwise, we must present the user with a set of alternatives that they can choose from. While similar formulations exist [8] [5], to our knowledge this is the first work to provide solutions for this multiobjective optimization problem.

2 Related work

This problem is similar to a Traveling Salesman Problem and its variations (see [4] for a comprehensive review), but varies in signficant ways. In this work, each site has a distinct probability of failure that differentiates it from other sites, imposing a preference order. More recent work extended the TSP to include features that assign value to visited locations [1] [11], but these models assume that costs are fixed and known. The Orienteering Problem with Stochastic Profits (OPSP) introduces a distribution over the profits at each site and seeks to meet a profit threshold within a given time limit [12].

Another key difference of this domain from prior work is the minimization of multiple competing objectives. Multi-objective optimization for path planning has been approached using evolutionary algorithms [6], but has often been limited to simple cycles on graphs. The Traveling Purchaser Problem with Stochastic Prices [8] introduces the model used in this work, but generates solutions by generalizing the cost distribution at each site as the expected cost, which we compare against in our evaluation in Section 6. More recent work has provided solutions for maximizing probability with a fixed budget and minimizing budget with a fixed probability [5] [2], but provides solutions for a path.

3 Problem formulation

We formulate the AS-ISR problem as a stochastic physical search problem (SPSP), where we are given an undirected network $G(S^+, E)$ with a set of sites $S^+ = S \cup \{o, d\}$ where $S = \{s_1, ..., s_m\}$ is the set of m sites that may be active, o and d are the origin and destination locations. We are also given a set of edges $E = \{(i, j)] : i, j \in S^+\}$. Each (i, j) has the cost of travel t_{ij} that is deterministic and known. An agent must start at origin node o, visit a subset of sites in S to find an active site, and then end at the destination point d.

The cost of identifying an ASAT at each site $s_i \in S$ is an independent random variable C_i with an associated probability mass function $P_i(c)$, which gives the probability that determining if a site is active will cost c at site s_i. This cost may be infinity, which indicates the site is not active and no information can be gained, or 0 to indicate there is no cost. We refer to this specialization as a *Binary SPSP*. A site s_i is active with some probability p_i, or inactive with probability $1 - p_i$, and the objective is to minimize the cost and probability of failure for visiting an active site.

A solution is a path π, a list of length n where $\pi(i)$ and $\pi(i+1)$ are consecutive locations along a path, with $\pi(1) = o$ and $\pi(n) = d$. We define $\pi(i, j)$ as the the sub-path of π containing consecutive path locations $\pi(i)$ through $\pi(j)$. In Figure 1 we show a small instance of an AS-ISR problem with 5 sites and the associated pareto set of path solutions. The longest solution path, the shortest path that visits all sites, is also the solution to the Traveling Salesman Path Problem [4].

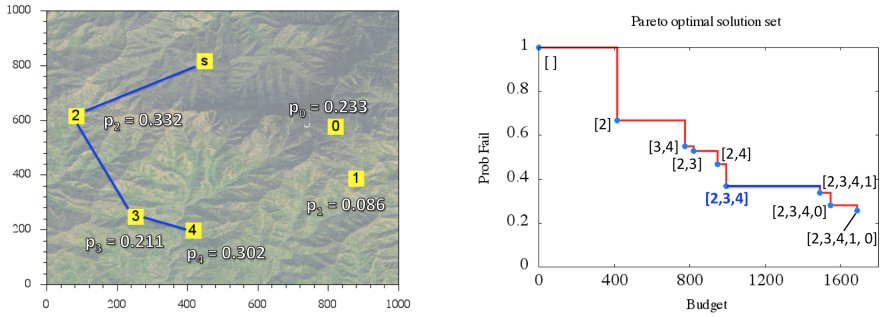

Fig. 1: An AS-ISR instance (left) with starting location s and 5 sites. Below each site is the probability that the site is active. The set of solutions (right) contains a collection of nondominated paths and their associated cost and probability. Each point is labeled with the path, which is the order of site visits starting from s. A single path (blue) is shown on both graphs, for reference.

4 Multiobjective optimization of SPSP

A multi-objective optimization problems (MOP) represents a trade-off between competing objectives. Efficient exploration of the solution space is difficult in the case where MOP criteria share dependencies on a common set of variables, such as budget and prob of failure, in this case. The following definitions are derived from multiobjective optimization literature [13], with adaptations for the specifics of the B-SPSP.

In general, a multi-objective problem is characterized by a vector of r decision variables, $x = (x_1, x_2, ..., x_r) \in X$, and k criteria. There is an evaluation function $F : X \to A$, where $A = (a_1, a_2, ...a_k)$ represents the k attributes of interest. We represent the fitness of vector x as $F(x) = (f_1(x), f_2(x), ...f_k(x))$, where $f_i(x)$ is the mapping from the decision variable vector to a single attribute a_i.

Definition 1.1 (Multiobjective Optimization for SPSP) We define a set of decision variables x denoting a path, a set of 2 objective functions for cost, $f_1 : X \to \mathbb{R}$, and probability of failure, $f_2 : X \to [0, 1]$, and a set of m constraints that require a decision vector be a valid path. We use probability of failure for clarity of definition, allowing us to minimize both objectives. The optimization goal is to

$$\begin{array}{ll} \text{minimize} & F(x) = (f_1(x), f_2(x)) \\ \text{subject to} & e(x) = (e_1(x), e_2(x), ..., e_m(x)) \leq 0 \\ \text{where} & x = (x_1, x_2, ..., x_n) \in X \end{array} \tag{1}$$

The *feasible set* X_f is defined as the set of decision vectors in X that form a valid path, with each decision vector x satisfying the path constraints $e(x)$, and such that $X_f = \{x \in X | e(x) \leq 0\}$. While we would prefer solutions that provide the minimum probability of failure at the minimum cost without violating the constraints, the objectives do not have optima that correspond to the same solution.

Definition 1.2. For a pair of objective vectors u and v,

$$\begin{array}{l} u = v \text{ iff } \forall i \in \{1, 2, ..., k\} : u_i = v_i \\ u \leq v \text{ iff } \forall i \in \{1, 2, ..., k\} : u_i \leq v_i \\ u < v \text{ iff } u \leq v \wedge u \neq v \end{array} \tag{2}$$

The relations $\geq$ and $>$ are defined similarly.

These definitions impose a partial order on solutions. When none of these relations hold between u and v, we can only state that $u \ngeq v$ and $u \nleq v$, meaning neither is superior. A solution that provides higher probability and higher cost than another solution is just as strong and may prove more effective if the additional budget is available to spend.

Definition 1.3 (Pareto Dominance) For any decision vectors b and c,

$$\begin{array}{ll} b \succ c \text{ (b dominates c)} & \text{iff } F(b) < F(c) \\ b \succeq c \text{ (b weakly dominates c)} & \text{iff } F(b) \leq F(c) \\ b \backsim c \text{ (b is indifferent to c)} & \text{iff } F(b) \not\leq F(c) \wedge F(b) \not\geq F(c) \end{array} \tag{3}$$

The definitions for "dominated by" $(\prec, \preceq, \backsim)$ are analogous.

An optimal solution is one that can not be improved by any other solution in the feasible set. This does not include solutions that are indifferent, as they are not comparable within the partial order imposed by pareto dominance. A decision vector $x \in X_f$ is nondominated with respect to a set $D \subseteq X_f$ iff $\nexists a \in D : a \succ x$. We refer to a solution x as Pareto optimal iff x is nondominated by X_f. Because the set of all solutions may be a partial order, there can be two or more Pareto optimal solutions for a given problem instance. The collection of all nondominated solutions with respect to the set of all solutions is referred to as the **nondominated solution set**.

Let $D \subseteq X_f$. The function $n(D) = \{d \in D : d \text{ is nondominated regarding } D\}$ outputs the set of nondominated decision vectors in D. The objective vectors in $f(n(D))$ is the *nondominated front* regarding D. The set $X_n = n(X_f)$ is referred to as the Pareto-optimal set (Pareto set). We use the term **solution set** to denote both nondominated decision set and their associated objective values. Solution sets represent the output of a solver that can be used to evaluate performance on problem instances.

4.1 Comparing solution sets

We desire a measure to compare solution sets, so that we can determine the trade-offs between using different solution techniques. Because the budget values are not normalized, and will vary greatly with respect to the number of sites, it is important that any measure be scale-independent. We use a measure similar to the $\mathcal{S}$ metric [13] on a two-dimensional space.

Definition 1.4 (Size of solution set) Let $D = (d_1, d_2, ..., d_m) \subseteq X$ be a set of decision vectors, ordered by f_1. The function $\mathcal{S}(D)$ returns the union of the area enclosed by the objective values for each vector d_i. The area enclosed by a single decision vector d_i is a rectangle defined by the points $(f_1(d_{i+1}), 0)$ and $(f_1(d_i), f_2(d_i))$. If $i = m$, then let $f_1(d_{i+1}) = f_1(d_i)$.

Using the $\mathcal{S}$ metric, we can derive a measure of error with respect to the optimal solution.

Definition 1.5 (Solution set error) Let $B^* = (x_1^*, x_2^*, ..., x_m^*) \subseteq X_f$ be the optimal set of decision vectors. Given a decision vector $D = (x_1, x_2, ..., x_m) \subseteq X$ we define the error of C as

$$\mathcal{E}(D) = \left(\frac{\mathcal{S}(D)}{\mathcal{S}(B^*)} \right) - 1 \tag{4}$$

A solution set should seek to minimize the $\mathcal{E}$ metric, with $\mathcal{E}(D) = 0$ indicating solution set D is optimal.

5 Approach

The Iterative Domination Solver (IDS) starts by generating all paths containing one market and adding them to the active set. Each iteration of of the algorithm consists of two phases: the *search phase* and the *domination phase*. In the search phase, remaining sites are added to all paths in the active set. This generates a new set of candidate paths that are added to the active set. In the domination phase, the solver evaluates new candidate paths and determines if they are dominated or dominate any other solutions. Depending on the filter depth parameter, dominated solution are removed from the active set, preventing any further search using that path. This procedure allows us to focus search on high value sub-paths without expending significant effort on paths that are low value and unlikely to be found on a non-dominated path.

5.1 Search Phase

A site is added to the previous path immediately after the start site or previous to the destination site, which ever adds lower total travel cost. This operation prevents insertions that break up edges from the last iteration, preserving inter-site edges that exist in sub-paths. An example of this operation is shown in Figure 2. This process continues, iteratively adding new markets to the candidate paths, until we have the path that visits all sites in the problem, at which point the process terminates and the non-dominated set of intervals is returned.

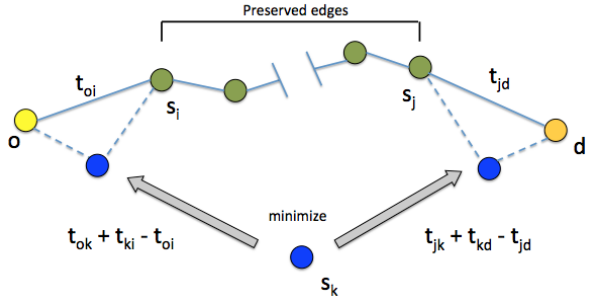

Fig. 2: Illustration of the insertion step for the IDS algorithm. A new node s_k is added to the search path after the origin site or before the destination site, whichever has the lower added path cost. This preserves the edges from the previous path in the search process, exploiting nondominated substructure.

We can reduce the search space via a *filter depth* (fd) parameter, which is path length at which we start pruning out dominated solutions. For example, if

we set $fd = 1$, only a single path consisting of (o, s_i, d) will remain after initial path generation, and all paths of length 2 must contain s_i. Then, if the only nondominated size 2 path contains s_i and s_j, all resulting paths from the next iteration must contain that ordered pair, and so on. Using filter depth is an effective pruning strategy because it prefers edges with shorter path lengths and there is a diminishing return on probability as the problem size increases.

5.2 Domination phase

A quad tree [3] is a rooted data structure where a node may have up to 4 children. For our purposes, each child represents a quadrant in the 2D plane of budget and probability of failure. This is defined recursively, making it an extension of a binary tree to 2 dimensions. The quad tree data structure allows us to store the value pairs (cost, probability) and their associated paths, while also detecting when a new solution is dominated or dominates another solution.

Because the number of possible paths is exponential with respect to the number of sites [10], storing all solutions is unreasonable. When a domination relation is detected, we use the structure to search for entries that should be removed or preserved and restructure the tree as necessary. This adds additional overhead during storage of individual solutions, but allows us to limit the space complexity of storing candidate solutions. The efficiency of searching this structure is sensitive to proper selection of the initial root node, an optimization still being investigated. In this work we choose to seed the tree with a random path using half of the markets, but better methods are certain to exist.

6 Experimental Results

We evaluate our approach using randomly generated problem instances and comparing performance against a number of intuitive approaches to solving the B-SPSP. Site networks are formed by placing sites randomly in a 1000×1000, with travel cost t_{ij} being the Euclidean distance between sites s_i and s_j. The probability of failure at each sites is drawn from a Gaussian distribution with varying mean and variance. All results are based on 100 instances, with error bars indicating a 95% confidence interval. We use a naming convention based on the chosen filter depth. If $fd = k$ then we refer to the algorithm as IDS-k.

In order to determine the relative performance of IDA, we compare against a solver for these problem types that minimizes Expected Cost (EC), adapted from the approach outlined in [8]. Changes to the algorithm are the acquisition of only a single 'item', and the removal of item cost, which is 0 in this domain.

At each step, EC will provide a path the minimizes the overall cost of the complete path, without consideration for budget or some fixed probability of success. In order to generate a solution set from this single path, we include all sub-paths of the solution beginning at the starting site. This gives us a step-wise set of intervals that still meet the criterion of minimizing expected cost. We also

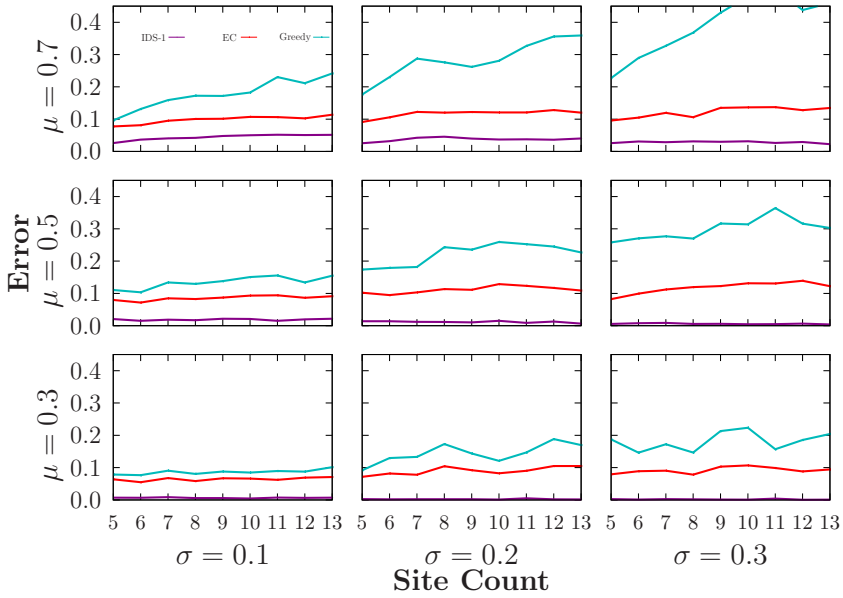

Fig. 3: Comparison of error on random instances with respect to mean and variance of probability of failure on sites

include a greedy solution that generates a set of solutions by selecting the lowest cost edge available.

Using the $\mathcal{E}$ metric defined in Section 4.1, we can compute the error of each approach with respect to optimal. In Figure 3, we show the effectiveness of IDS with respect to both changes in the mean and variance of the probability of failure. As the mean probability of failure increases, all 3 algorithms perform closer to optimal. This is due mostly to each site individually moving closer to 0% failure, leaving less room for error, in general.

As the variance on the probability is increased, expected cost seems relatively unaffected, the greedy solver performs significantly worse, and IDS sees significant improvement. Increased variance distinguishes the site probabilities, making higher cost edges feasible options for paths that may be non-dominated. This makes the single path with low average cost less likely to cover the space of non-dominated solutions.

In Figure 4 we show the execution time of IDS with expected cost, optimal and a random solver that generates 30,000 randomly selected paths. IDS completes search orders of magnitude faster than optimal and EC, mainly due to being able to terminate search before expanding the search tree on paths that are not of good quality.

Because the optimal solution requires exponential time to solve as the problem size increases, it is difficult to compare performance against optimal for large instances. To show that the performance of IDS is maintained for large problem

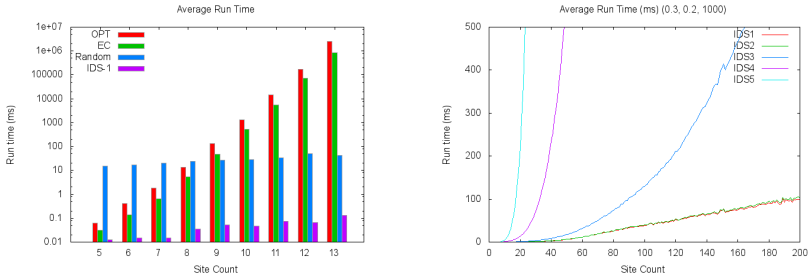

Fig. 4: Comparison of average execution time among solvers (left) and execution time for various IDS filter depths (right).

instances, we compare against a random solver, which generates 30,000 random paths and generates a solution set from all enumerated sub-paths. We define the lift of set D as $\left(\frac{\mathcal{S}(D)}{\mathcal{S}(R)}\right) - 1$, where R is the nondominated solution set generated by the random solver. The results of this analysis are shown in Figure 5. While increasing the filter depth provides minor gains in performance, the increased computational demand is significant, especially for large instances.

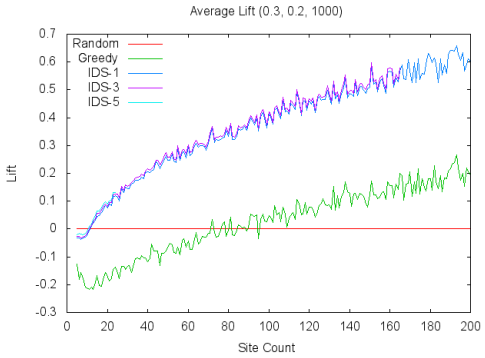

Fig. 5: Lift of IDS and greedy versus random sampling

7 Conclusions and Future Work

We have presented B-SPSP as a method for solving AS-ISR problems, providing analysis of the problem space and a characterization of nondominated solution sets. We have also presented the Iterative Domination Solver as an approximate method for generating solution sets. We have shown that IDS performs better than existing approaches across a wide range of problem parameters with orders

of magnitude less execution time on randomly generated instances. While the filter depth can be increased for minor performance gains on instances of the AS-ISR problem, in most cases IDS-1 will prove sufficient.

Future work will focus on solution sets for general SPSP instances with multiple costs per site, which will significantly increase the complexity of finding optimal solutions due to the branching of search on cost realizations at each site. We plan to explore how IDS can be adapted to solve these instances using the same iterative approach with pruning, and we expect to see larger gains from increased filter depth. Additionally, a number of solution concepts have been developed for TSP that could, in some cases, be adapted for this problem formulation.

References

1. Arora, S., Karakostas, G.: A 2+ ε approximation algorithm for the k-mst problem. In: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms. pp. 754–759. Society for Industrial and Applied Mathematics (2000)
2. Brown, Daniel, S., Hudack, J., Banerjee, B.: Algorithms for stochastic physical search on general graphs (to appear) (2015)
3. Finkel, R.A., Bentley, J.L.: Quad trees a data structure for retrieval on composite keys. Acta informatica 4(1), 1–9 (1974)
4. Gutin, G., Punnen, A.P.: The traveling salesman problem and its variations, vol. 12. Springer (2002)
5. Hazon, N., Aumann, Y., Kraus, S., Sarne, D.: Physical search problems with probabilistic knowledge. Artificial Intelligence 196, 26–52 (Mar 2013)
6. Jozefowiez, N., Glover, F., Laguna, M.: Multi-objective Meta-heuristics for the Traveling Salesman Problem with Profits. Journal of Mathematical Modelling and Algorithms 7(2), 177–195 (Feb 2008), http://link.springer.com/10.1007/s10852-008-9080-2
7. Kambhampati, S.: Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain models. Proceedings of the National Conference on Artificial . . . pp. 1601–1604 (2007)
8. Kang, S., Ouyang, Y.: The traveling purchaser problem with stochastic prices: Exact and approximate algorithms. European Journal of Operational Research 209(3), 265–272 (Mar 2011)
9. Nguyen, T.A., Do, M., Gerevini, A.E., Serina, I., Srivastava, B., Kambhampati, S.: Generating diverse plans to handle unknown and partially known user preferences. Artificial Intelligence 190, 1–31 (Oct 2012)
10. Roberts, B., Kroese, D.P.: Estimating the Number of s - t Paths in a Graph Counting by Estimation 11(1), 195–214 (2007)
11. Snyder, L.V., Daskin, M.S.: A random-key genetic algorithm for the generalized traveling salesman problem. European Journal of Operational Research 174(1), 38–53 (2006)
12. Tang, H., Miller-Hooks, E.: A tabu search heuristic for the team orienteering problem. Computers & Operations Research 32(6), 1379–1407 (2005)
13. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization : Methods and Applications (30)