Column Generation Based Heuristics for a Generalized Location Routing Problem with Profits Arising in Space Exploration

Jaemyung Ahn (jaemyung@mit.edu)
Olivier de Weck (deweck@mit.edu)
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology

Yue Geng (yue-geng@northwestern.edu)
Diego Klabjan (d-klabjan@northwestern.edu)
Department of Industrial Engineering and Management Sciences
Northwestern University

January 15, 2012

Abstract

The generalized location routing problem with profits is a new routing problem class that combines the vehicle routing problem with profits, depot or base selection, and the notion of strategies and route tactics to be employed at selected bases. The problem simultaneously determines base locations, strategies to use at the bases, sites to visit, and routes to carry out the visits, to maximize the sum of profits that can be obtained by visiting sites, under resource consumption constraints. The problem arises in exploration of planetary bodies where strategies correspond to different technologies. A description of the generalized location routing problem with profits and its mathematical formulation as an integer program are provided. Two solution methodologies to solve the problem - branch-and-price and a three-phase heuristic method combined with a generalized randomized adaptive search procedure - are proposed. Numerical experiments for the two solution methodologies are carried out. Their performance in terms of computation time and optimality gap are analyzed and compared to establish the problem characteristics, indicating which method is more advantageous.

1 Introduction

During exploration of planetary bodies, NASA first identifies a set of potential surface locations to serve as exploration bases. At each base various strategies can be used. Among all strategies, exactly one has to be employed at each selected base. Each base can have its own strategies and they do not have to be the same for all bases. For example, the standard strategy might imply that once an exploration vehicle leaves the base, the route length must not exceed its fuel capacity. On the other hand, under the orbiting depot strategy, a depot might orbit around the planetary body and it can provide additional fuel to the exploration vehicles. If this strategy is employed at a base, routes can be longer. The selected strategy at a base implies a set of resource constraints on the routes originating at the base.

Exploration sites are potential locations on the surface to be explored by NASA scientists and astronauts. All sites are not required to be visited, however, exploring a site yields a given

profit (scientific value). A selected site must be explored from exactly one route originating and finishing at the same base (and complying to the selected strategy at the base).

All site explorations must be performed within a predefined budget, which captures resource consumption on routes and the direct cost associated by using a strategy. NASA is faced with a daunting task to simultaneously determine base (depot) locations, strategies to use at the bases, sites to visit, and routes to carry out the visits, to maximize the sum of the profits that can be obtained by visiting sites, under resource consumption constraints. This problem also arises during military surface operations, and recruiting situations where constraints do not allow to visit all sites of interest.

In a more traditional logistics setting, the problem is to select several depots from potential locations. At each depot, there are potential modes of transportation to be employed. A selected mode at a depot implies resource constraints on routes originating at the depot (e.g., vehicle capacity, travel times between two customers, etc.). Not all customers are required to be visited, however, each visited customer yields a certain profit and thus the problem is to maximize the total profit. On the other hand, deploying a mode of transportation at a depot requires a certain cost. A solution must select strategies that are within a predefined budget.

The location routing problem (LRP), one of the existing routing problem classes, is closest to the problem discussed in this paper. The LRP determines depot (base, facility) locations out of a candidate set and feasible vehicle routes to visit sites (customer locations) associated with the selected depots, to minimize the sum of costs related to depot selection and/or total travel distance. In the classic LRP each site must be visited exactly once and some side constraints (vehicle capacity, route length, visiting time windows) are typically imposed. The LRP is a very complex problem where various heuristic methods (e.g., Srivastava and Benton (1990), Tuzun and Burke (1999), Duhamel et al. (2010)) and exact methods (e.g., Laporte et al. (1988, 1989), Berger et al. (2007), Karaoglan et al. (2011)) are proposed. Min et al. (1998) provide a survey and explore promising research opportunities. Nagy and Salhi (2007) is a more recent survey that classifies LRP papers based on hierarchical structure, type of input data, planning period, solution method, etc.

While conventional routing problems (the traveling salesman problem (TSP), the vehicle routing problem (VRP), and the LRP) impose a restriction that each site must be visited exactly once, there are situations in which all sites cannot be visited because of constraints. These problems must be treated differently. Let us assume it is possible to quantify the profit that can be obtained by visiting each site. Instead of visiting every site, this restriction is relaxed and visits to only selected sites are admissible. The problem is to optimize an objective function reflecting the profits obtained by the visits, under resource constraints. There are some studies on the extensions of the TSP and VRP corresponding to this relaxation (e.g., Tsiligirides (1984), Kataoka and Morito (1988), Balas (1989), Laporte and Martello (1990), Butt and Cavalier (1994), Dell'Amico et al. (1988), Chao et al. (1996), Butt and Ryan (1999), Gueguen (1999), Tang and Miller-Hooks (2005), Feillet et al. (2005), Bérubé et al. (2009)). While there are several different versions of this extension, the most relevant problems are those generated by an extension from the TSP known as

the *traveling salesman problem with profits* (TSPP) and those by an extension of the VRP called the *vehicle routing problem with profits* (VRPP). A well studied subclass of VRPP is the *team orienteering problem* (TOP), where the goal is to determine a given number of paths, each limited by a given resource upper bound, that maximize the total profits collected. Vansteenwegen et al. (2011) is a recent survey on the orienteering problem (a subclass of TSPP) and TOP, and different heuristics are proposed to solve such problems (e.g., Souffriau et al. (2011), Archetti et al. (2011)).

Figure 1 presents a *lineage tree* of the routing problem family. The classical TSP, which is the simplest routing problem, is located in the top-left part of the tree. We can see two types of evolutions in the tree - vertical and horizontal evolutions. The vertical evolution in the lineage tree makes the problem more and more complex by allowing multiple routes, multiple bases, and base location selection. The horizontal evolution relaxes the restriction that each site must be visited.

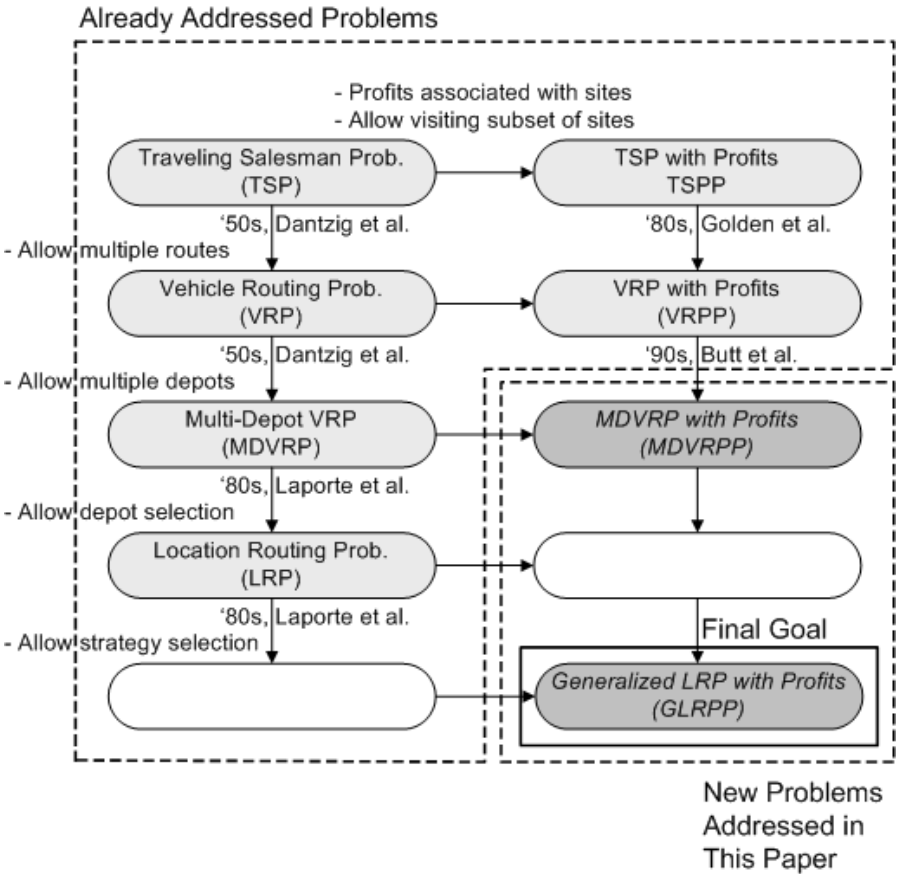

Figure 1: Lineage Tree of the Routing Problem Family

The problem presented in this paper combines the LRP and routing problem with profits (horizontal evolution from LRP) and adds a new feature - strategies for selected bases (vertical evolution). We refer to the problem as the generalized location routing problem with profits (GLRPP). The GLRPP is a new problem class which arose in the context of NASA planetary

surface exploration (NASA and other space agencies (2007)). Beyond surface exploration campaigns, Ahn et al. (2008) discusses other applications such as college football recruiting of an NFL team, drilling of exploration wells in the oil/gas industry, and military operations over multiple locations as potential applications of the GLRPP framework.

We model the GLRPP as a large-scale mixed integer program (IP). Due to an excessive number of variables, delayed column generation is employed for solving LP relaxations. To find integral solutions, a branch-and-price algorithm is applied. By using diving, we either stop upon finding the first integral solution or detecting infeasibility of the LP relaxation. In either case, at the end we solve an IP by considering all the columns generated during the entire branch-and-price procedure. This will guarantee feasibility since visiting none of the sites by using no route is obviously a feasible solution. We also design a simpler but novel heuristic based on decomposition, randomized greedy local search, and integer programming. We first decompose the network into clusters. Next we apply a greedy randomized adaptive search procedure (GRASP) to select a subset of considered bases and strategies for each cluster. Then we compute the optimal routes based on selected strategies, bases, and cluster by price-and-branch. Finally, a budget feasible set of strategies is selected by solving an auxiliary IP.

The main contributions of our work are:

1. introduction and modeling of a new routing problem with a novel application in space exploration, which include the notion of strategies employed at bases; the model generalizes and combines two separate threads in vehicle routing,
2. a heuristic branch-and-price algorithm, relying on tailored branching,
3. a novel heuristic combining price-and-branch, GRASP, and integer programming,
4. the study of a real-world case encountered in space exploration.

This paper is structured as follows. Section 2 develops a mathematical formulation of the GLRPP. Section 3 discusses the two solution methodologies to solve the GLRPP. Results of numerical experiments, including the comparison of the methods, are presented in Section 4. Finally, Section 5 provides concluding remarks. The nomenclature is provided in Appendix A.

2 Mathematical Formulation of the GLRPP

In this section, a brief description and mathematical formulation of the GLRPP are provided. Since our primary application is in space exploration, we use the related terms instead of traditional logistics terms (see the introduction for further discussions). Consider a complete graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} = \mathbf{B} \cup \mathbf{E}$ is the index set of nodes and $\mathcal{A} = \{(i,j) | i \in \mathcal{N}, j \in \mathcal{N}\}$ is the set of all arcs. Set $\mathbf{B} = \{1, \ldots, n_B\}$ is the index set of potential bases (n_B is the number of potential bases) and set $\mathbf{E} = \{n_B + 1, \ldots, n_B + n_E\}$ is the index set of sites (n_E is the number of sites). Typically n_E is larger than n_B. For each potential site $i \in \mathbf{E}$, two values v_i and t_i are assigned; v_i represents the profit that can be obtained by visiting site i, and t_i is the time required to

collect the profit. Cost (typically a function of distance) associated with each arc is expressed as a matrix $\mathbf{C} = [c_{ij}]$ where c_{ij} represents the cost of the arc from i to j ($i \in \mathcal{N}, j \in \mathcal{N}$). We assume that the triangle inequality holds for $\mathbf{C}$.

A route related to a potential base $b \in \mathbf{B}$ is a sequence of nodes representing the transportation among nodes. Potential base b starts the route, it is followed by some sites from $\mathbf{E}$, and it again ends the route. A mission associated with base b is a set of routes related to b. A campaign is a set of all missions, which are defined over graph $\mathcal{G}$ and are collectively pursuing an objective. A mission associated with base b must also specify a unique strategy, which bears a certain cost. With each strategy at a base, a set of routing tactics must be selected, one for each route assigned to the base. A routing tactic determines feasibility of the underlying route.

Example. Hereafter we consider a possible exploration case of Mars, which comes from NASA. Potential landing locations (bases) and exploration sites on the Mars surface are pre-determined. Once the astronauts land on Mars, they build a base at the landing site and visit scientifically interesting sites to gather scientific information using surface exploration vehicles. The profit assigned to a site is the scientific return that can be achieved by visiting the site. A route to visit a subset of sites starts from and terminates at the base. Based on the existing technologies, there are two available strategies.

Standard strategy: Under the standard strategy, astronauts explore the Mars surface using a surface exploration vehicle. Loading capacity of a surface exploration vehicle is limited and a vehicle has to return to the base before it runs out of fuel.

Orbiting depot strategy: There is an alternative possible strategy called the orbiting depot strategy. If this strategy is employed at the base, an orbiting fuel depot circulates around Mars and provides additional fuel to the exploration vehicles. For each route, there are two tactics to consider based on fueling. The standard tactic is to take all of the fuel with the vehicle and thus it has identical restriction as the routes under the standard strategy. An alternative is to consider the depot-assisted tactic at a route. In the depot-assisted tactic, a supply unit from the orbiting depot circulating around Mars provides additional fuel, which effectively doubles the fuel capacity of the vehicle. In turn, the vehicle can double the traveling distance. Thus bases with the orbiting depot strategy have one more option regarding routes.

Figure 2 depicts the strategies and underlying route tactics. □

Given a base and set of sites belonging to a route, only a minimum distance path that can be obtained by solving the TSP is considered as a viable route; we do not consider inefficient routes. Thus the total number of routes related to a base, regardless of the feasibility of routes, equals the total number of subsets of site set $\mathbf{E}$, i.e., 2^{n_E}. We define $J = \{0, \ldots, 2^{n_E} - 1\}$ as the index set of subsets of site set $\mathbf{E}$.

Let $\mathbf{S}$ be the set of all possible mission strategies. Each mission strategy implies a set of routing tactics that govern route feasibilities and resource consumptions along the routes. Each

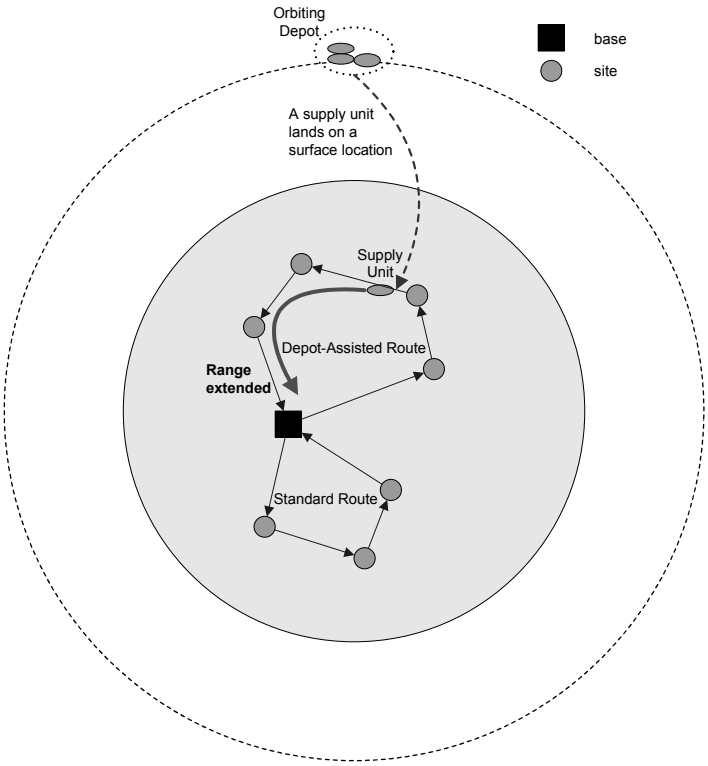

Figure 2: The Two Strategies and the Corresponding Tactics

selected base can have its own mission strategy. Routing tactic $k \in \mathbf{T}^s$, where $\mathbf{T}^s = \{1, 2, \ldots, t^s\}$ is the index set of routing tactics available for mission strategy $s \in \mathbf{S}$, characterizes a route by specifying single-route constraints and a maximum number of routes. Single-route constraints are expressed by resource types, resource consumption coefficient vectors ($\mathbf{c}^{s,k}$, $\bar{\mathbf{c}}^{s,k}$, $\tilde{\mathbf{c}}^{s,k}$), and a resource consumption limit vector $\mathbf{u}^{s,k}$ for the constraining resources. The first coefficient $\mathbf{c}^{s,k}$ is the fixed usage of the resource, the second one specifies the per-arc cost resource usage, and $\tilde{\mathbf{c}}^{s,k}$ the per-node time usage. The maximum number of routes $n^{s,k}$ limits the number of routes using tactic $k \in \mathbf{T}^s$ included in a mission.

Example. We have $\mathbf{S} = \{\text{standard}, \text{orbiting}\}$ and $\mathbf{T}^{\text{standard}} = \{\text{standard}\}$, $\mathbf{T}^{\text{orbiting}} = \{\text{standard}, \text{orbit-assisted}\}$. In our case $\tilde{\mathbf{c}}^{s,k}$ is the same for both strategies and routing tactics, however $\mathbf{u}^{\text{orbiting,orbit-assisted}} = 2 \cdot \mathbf{u}^{\text{standard,standard}} = 2 \cdot \mathbf{u}^{\text{orbiting,standard}}$.

Under the standard tactic, we can have an unlimited number of routes since bringing fuel on the surface is not an issue. On the other hand, technological limitations restrict at most 4 orbiting supply units and thus for each base the maximum number of routes with orbit-assisted tactic is 4. □

Set $J^{b,s,k} \subset J$ is the index set of feasible routes related to base b with respect to routing tactic

k of mission strategy s and is defined as follows:

$$J^{b,s,k} = \{j \in J | \underbrace{\mathbf{c}^{s,k}}_{per-route} + \underbrace{\mathrm{TSP}_j^b \cdot \bar{\mathbf{c}}^{s,k}}_{on-arc} + \underbrace{(\sum_{i \in \mathbf{R}_j} t_i) \cdot \tilde{\mathbf{c}}^{s,k}}_{on-site} \leq \mathbf{u}^{s,k}\}, \quad (1)$$

where $\mathbf{R}_j$ is the corresponding set of sites on route j and TSP_j^b is the travel distance of the TSP solution on nodes $\{b\} \cup \mathbf{R}_j$.

Mission strategy $s \in \mathbf{S}$ characterizes a mission by specifying a set of available routing tactics $\mathbf{T}^s$, collective constraints, and mission cost C^s. Collective constraints are expressed by resource types, resource consumption coefficient vectors ($\mathbf{d}^s$, $\bar{\mathbf{d}}^s$, $\tilde{\mathbf{d}}^s$), and a resource consumption limit vector $\bar{\mathbf{u}}^s$ for the constraining resources. Each route consumes a fixed amount $\mathbf{d}^s$, the travel distance weighted by $\bar{\mathbf{d}}^s$, and the time spent to collect profit weighted by $\tilde{\mathbf{d}}^s$.

Example. The cost associated with a strategy is measured by the total mass that should be sent to the Mars surface when the strategy is used. The cost for the orbiting depot strategy is larger than that of the standard strategy. The difference results from the mass of additional supply units.

Each vehicle has a limited life span. A vehicle has to return to the Earth after a certain amount of time. This implies collective constraints with time being the resource. Coefficients $\tilde{\mathbf{d}}^s$ and $\bar{\mathbf{d}}^s$ encode the time spent at a site and the travel time between two locations. A possible fixed time to prepare the vehicle for exploration is captured by $\mathbf{d}^s$. The collective coefficient $\bar{\mathbf{u}}^s$ imposes the time limit a vehicle can spend on the surface over all of its routes. In our example we have $\bar{\mathbf{u}}^{\text{standard}} = \bar{\mathbf{u}}^{\text{orbiting}}$ since the lifespan of the two vehicles is the same. □

We assume all missions in a single campaign are funded under the same budget source. There is a budget constraint on the cost sum for all missions in a campaign; the cost sum cannot exceed a pre-determined budget M. Figure 3 shows the hierarchical structure of decisions and parameters for the GLRPP.

The objective of the campaign is to maximize the sum of profits that can be obtained during the campaign. Decision variables are selection of bases that are used, mission strategies for the selected bases, selection of routes to visit sites from each of the selected bases, and routing tactics for the selected routes. The solution of the problem should satisfy single-route constraints (as listed in (1)), maximum route number constraints, collective constraints, and the campaign budget constraint. A constraint that a single site cannot be visited multiple times should also be satisfied.

Figure 4 presents a sample GLRPP instance. Potential bases and sites are expressed as squares and circles, respectively. The profit that can be obtained by visiting each site is proportional to the diameter of the circle representing the site, and is also presented as the number beside the circle. Out of three potential bases, one base is used with strategy 1 (orbiting) that has two routing tactics (orbit-assisted and standard), another base is used with strategy 2 (standard) that has only one routing tactic (standard), and the last potential base is unused. The solution

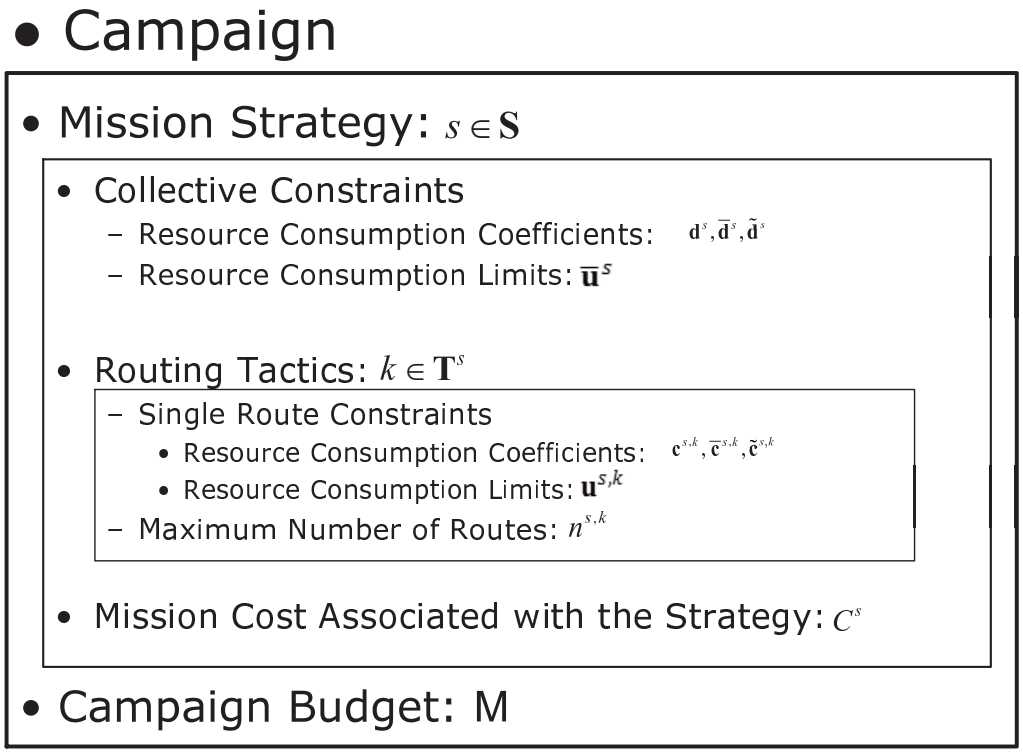

Figure 3: Decision and Parameter Hierarchy for the GLRPP

satisfies all resource constraints imposed on every single route, mission, and the whole campaign, which have not been explicitly exhibited in the figure.

Next we formulate the GLRPP as an IP. The model GLRPP reads as follows.

$$\max_{x_j^{b,s,k}, y^{b,s}} \sum_{b\in\mathbf{B}} \sum_{s\in\mathbf{S}} \sum_{k\in\mathbf{T}^s} \sum_{j\in J^{b,s,k}} f_j \cdot x_j^{b,s,k} \tag{2}$$

$$\sum_{b\in\mathbf{B}} \sum_{s\in\mathbf{S}} \sum_{k\in\mathbf{T}^s} \sum_{j\in J^{b,s,k}} \mathbf{A}_j \cdot x_j^{b,s,k} \le \mathbf{1}_{n_E} \tag{3}$$

$$\sum_{k\in\mathbf{T}^s} \sum_{j\in J^{b,s,k}} \mathbf{h}_j^{b,s} \cdot x_j^{b,s,k} \le \bar{\mathbf{u}}^s \cdot y^{b,s} \qquad b \in \mathbf{B}, s \in \mathbf{S} \tag{4}$$

$$\sum_{j\in J^{b,s,k}} x_j^{b,s,k} \le n^{s,k} y^{b,s} \qquad b \in \mathbf{B}, s \in \mathbf{S}, k \in \mathbf{T}^s \tag{5}$$

$$\sum_{s\in\mathbf{S}} y^{b,s} \le 1 \qquad b \in \mathbf{B} \tag{6}$$

$$\sum_{b\in\mathbf{B}} \sum_{s\in\mathbf{S}} C^s y^{b,s} \le M \tag{7}$$

$$x_j^{b,s,k} \in \{0,1\} \qquad b \in \mathbf{B}, s \in \mathbf{S}, t \in \mathbf{T}^s, j \in J^{b,s,k} \tag{8}$$

$$y^{b,s} \in \{0,1\} \qquad b \in \mathbf{B}, s \in \mathbf{S} \tag{9}$$

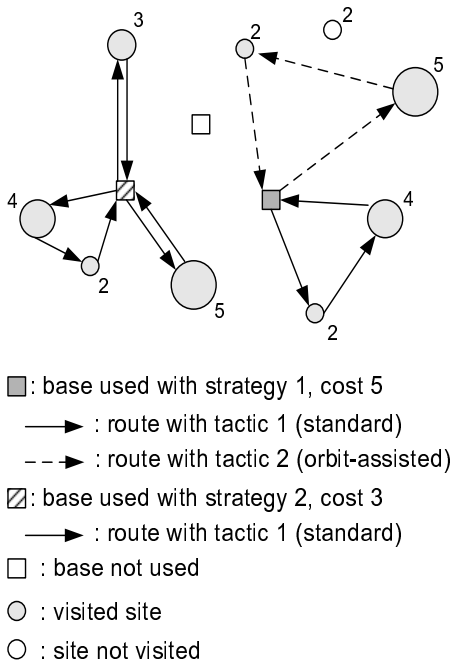

Figure 4: Sample GLRPP Instance and Solution

Here $\mathbf{1}_k$ denotes the k-dimensional vector with all 1's. The decision variables are $x_j^{b,s,k}$ and $y^{b,s}$. Variable $x_j^{b,s,k}$ is 1 if the route determined by base b and site subset $\mathbf{R}_j$ using routing tactic k and mission strategy s is included in the solution, and is 0 otherwise. Variable $y^{b,s}$ is 1 if mission strategy s is selected by base b, and is 0 otherwise. Profit f_j is the sum of profits for all sites belonging to $\mathbf{R}_j$ and equals $\sum_{i \in \mathbf{R}_j} v_i$. Constraint (3) requires that each site is visited no more than once. Entry a_{ij} of A_j is 1 if site $i \in \mathbf{E}$ is in $\mathbf{R}_j$ and 0 otherwise. Collective constraints (4) impose that the resource sum for routes in a mission is bounded. Vector $\mathbf{h}_j^{b,s}$ represents consumption of constraining resource types for collective constraints specified by mission strategy s on the route determined by base b and site subset $\mathbf{R}_j$ and is defined as follows:

$$\mathbf{h}_j^{b,s} = \underbrace{\mathbf{d}^s}_{per-route} + \underbrace{\mathrm{TSP}_j^b \cdot \bar{\mathbf{d}}^s}_{on-arc} + \underbrace{(\sum_{i \in R_j} t_i) \cdot \tilde{\mathbf{d}}^s}_{on-site}. \tag{10}$$

Constraints (5) require that for each base, the number of routes using routing tactic k of mission strategy s be at most $n^{s,k}$. Constraints (6) impose that no more than one strategy is selected at a base. If no strategy is selected for base b ($y^{b,s} = 0$ for all $s \in \mathbf{S}$), then base b is not used. Constraint (7) imposes that the sum of costs for all missions in a campaign does not exceed budget M.

The matrix version formulation of the GLRPP is written as

$$\max_{\mathbf{x},\mathbf{y}} \quad \mathbf{rx}$$

$$\mathbf{Ax} \leq \mathbf{1}_{n_1} \tag{11}$$

$$\mathbf{Hx} - \mathbf{Ly} \leq \mathbf{0}_{n_2} \tag{12}$$

$$\mathbf{E}_1\mathbf{x} - \mathbf{Ny} \leq \mathbf{0}_{n_3} \tag{13}$$

$$\mathbf{E}_2\mathbf{y} \leq \mathbf{1}_{n_4} \tag{14}$$

$$\mathbf{cy} \leq \mathrm{M} \tag{15}$$

$$\mathbf{x}, \mathbf{y} \text{ binary.}$$

Vectors $\mathbf{x}$ and $\mathbf{y}$ are decision variable vectors for the GLRPP, which are composed of decision variables in (8) and (9). Matrices $\mathbf{A}$, $\mathbf{H}$, $\mathbf{L}$, $\mathbf{E}_1$, $\mathbf{N}$, $\mathbf{E}_2$, and vectors $\mathbf{r}$, $\mathbf{c}$ are created from the coefficients in constraints (3)-(7). Values n_1, n_2, n_3, and n_4 represent the numbers of rows in constraints (11), (12), (13), and (14): $n_1 = n_E$ equals the number of sites; $n_2 = n_B \cdot |S|$ represents the total number of collective constraints for all potential bases using all possible mission strategies; $n_3 = n_B \cdot \sum_{s \in \mathbf{S}} |\mathbf{T}^s|$ is the total number of routing tactics for all potential bases using all possible mission strategies; and $n_4 = n_B$ is the total number of potential bases.

3 Solution Methodologies

Two different solution methodologies for the GLRPP - a heuristic *branch-and-price* algorithm and a heuristic *three-phase method* - are proposed in this section. We first discuss the branch-and-price approach.

3.1 Branch-and-Price

Branch-and-price is a branch-and-bound algorithm where LP relaxations are solved by delayed column generation. Usually problem specific branching strategies are employed. Due to computational time restrictions, we do not exploit the entire branch-and-bound tree. Instead we dive along a single branch of the tree.

In the remainder of the section we present details for solving the root node LP relaxation using column generation and obtaining a near-optimal solution.

3.1.1 The Dual Problem

The dual GLRPPLRD of the LP relaxation of GLRPP reads

$$\min_{\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_4,\mathrm{p}_5} \mathbf{p}_1'\mathbf{1}_{n_1} + \mathbf{p}_4'\mathbf{1}_{n_4} + \mathrm{p}_5 \cdot \mathrm{M}$$

$$\mathbf{p}_1'\mathbf{A} + \mathbf{p}_2'\mathbf{H} + \mathbf{p}_3'\mathbf{E}_1 \geq \mathbf{r} \tag{16}$$

$$-\mathbf{p}_2'\mathbf{L} - \mathbf{p}_3'\mathbf{N} + \mathbf{p}_4'\mathbf{E}_2 + \mathrm{p}_5\mathbf{c} \geq \mathbf{0} \tag{17}$$

$$\mathbf{p}_1 \geq \mathbf{0}, \mathbf{p}_2 \geq \mathbf{0}, \mathbf{p}_3 \geq \mathbf{0}, \mathbf{p}_4 \geq \mathbf{0}, \mathrm{p}_5 \geq 0.$$

Vectors $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$, and value p_5 are dual variables associated with constraints (11)-(15), respectively. Constraints (16) and (17) are related to $\mathbf{x}$ and $\mathbf{y}$, respectively. Next, we discuss the column generation method to solve the LP relaxation of the GLRPP.

3.1.2 Solving the LP Relaxation by Column Generation

Although binary requirements imposed on the decision variables for the GLRPP are relaxed in the LP relaxation, it is still prohibitively difficult to solve it because of the large number of decision variables. The column generation method is used to solve it without enumerating all columns (Barnhart et al. (1998), Desaulniers et al. (2005)).

In every iteration of the column generation procedure, we create an LP, called the restricted master problem, using columns that have been identified so far. We obtain optimal values of the primal and dual variables for the restricted master problem. Using these optimal dual values, we identify new columns that are added to the current set of columns in the restricted master problem. After the new restricted master problem is created, it is again solved and the procedure is repeated. We stop iterating when we can no longer identify a column that would improve the objective value.

Since the number of y variables is polynomial, they are always included in the restricted master problem. As a result, the dual solution to the restricted master problem always satisfies (17). The reduced cost of variable $x_j^{b,s,k}$ corresponding to (16) reads

$$r_j^{b,s,k} = \sum_{i \in R_j} \Big((p_1')_i + (p_2')^{b,s} \cdot \tilde{\mathbf{d}}^s \cdot t_i - v_i \Big) + (p_3')^{b,s,k} + (p_2')^{b,s} \cdot \mathbf{d}^s + \mathrm{TSP}_j^b \cdot (p_2')^{b,s} \cdot \bar{\mathbf{d}}^s.$$

The pricing problem, which is to find variables to add to the restricted master problem, is the following optimization problem:

$$\min_{\substack{b \in \mathbf{B}, s \in \mathbf{S} \\ k \in \mathbf{T}^s, j \in J^{b,s,k}}} r_j^{b,s,k}. \tag{18}$$

For a fixed b, s, k, this problem resembles the prize-collecting TSP, but it is more general since the objective function includes both node and edge weights. The latter are hidden within TSP_j^b.

A common way to solve (18) is by elementary constrained shortest path. The underlying graph can have negative cost cycles and thus only elementary paths should be considered. Such algorithms are very complex and hard to implement. For these reasons and since the anticipated

routes should not include too many nodes, we chose to adapt the approach from Butt and Ryan (1999).

Butt and Ryan (1999) present a pricing algorithm for a special case of (18) where b, s, k are not present and the reduced cost does not include TSP_j^b. The latter requires a modification of their pricing scheme, which is presented next.

We start by outlining the algorithm from Butt and Ryan (1999) under the assumption that b, s, k are fixed and there is no TSP_j^b in the reduced cost. For every i we define $q_i^{b,s} = (p_1')_i + (p_2')^{b,s} \cdot \tilde{\mathbf{d}}^s \cdot t_i - v_i$. Since the pricing problem is a minimization problem, sites with low $q_i^{b,s}$ are desirable to be included in a route. It is easy to see that due to the triangle inequality each site i with $q_i^{b,s} \geq 0$ is not going to be included in a route and thus we can neglect such sites. Without loss of generality let us assume that $q_1^{b,s} \leq q_2^{b,s} \leq \cdots \leq q_{Q^{b,s}}^{b,s} < 0$ for $Q^{b,s} \leq n_E$, and for each remaining site i we have $q_i^{b,s} \geq 0$. The algorithm greedily constructs a subset of sites to visit based on this order. For the incumbent subset of sites, its reduced cost is computed. If it is negative, then the TSP problem is solved on the underlying subset. If the resource constraint based on the TSP tour is not violated, the corresponding variable is selected for inclusion in the restricted master problem and the next site based on the order is added to the current set. If either the reduced cost is nonnegative or the TSP tour violates the resource constraint, we backtrack by removing the site added last to the subset.

The algorithm can also be described by using the notion of the lexicographical order where we consider subsets of sites as binary characteristic vectors with $Q^{b,s}$ coordinates. If vector Z corresponds to the current subset (its binary encoding), then adding a new site is equivalent to setting Z to the next vector in the lexicographical order. When removing the last site added from the set and adding the next one, we simply set Z to be the maximum vector $\bar{Z}$ smaller than Z based on the lexicographical order with $|\bar{Z}| \leq |Z|$ ($|Z|$ is the number of sites in the underlying set, or, in other words, the number of bits at 1).

We now present the necessary changes to solve (18). We first provide the following lower bound on the reduced cost. For $i \in \mathcal{N}$ let $u_i = \min_{i_1} c_{i,i_1}$ and let

$$l_j^{b,s,k} = \sum_{i \in R_j} \left((p_1')_i + (p_2')^{b,s} \cdot \tilde{\mathbf{d}}^s \cdot t_i - v_i + (p_2')^{b,s} \cdot \bar{\mathbf{d}}^s \cdot u_i \right) + (p_3')^{b,s,k} + (p_2')^{b,s} \cdot \mathbf{d}^s + u_b \cdot (p_2')^{b,s} \cdot \bar{\mathbf{d}}^s \,.$$

Then it is easy to see that $r_j^{b,s,k} \geq l_j^{b,s,k}$. This follows from $\text{TSP}_j^b \geq \sum_{i \in R_j} u_i + u_b$.

The complete algorithm is detailed in Algorithm 1. Every Z encodes a route j and thus, for example, the quantity $r_Z^{b,s,k}$ represents the reduced cost of the route associated with the binary encoding Z. It is easy to see that due to the triangle inequality any site i with nonnegative $q_i^{b,s}$ is not going to be included in the lowest reduced cost subset. Thus we retain the same definition of $q_i^{b,s}$ and $Q^{b,s}$. In our case we need to first loop over all bases b and strategies s since the order depends on them. This minimizes the number of TSP evaluations in step 7. Note that if $l_Z^{b,s,k} \geq 0$ for all k, then the resulting reduced cost of the route corresponding to Z is also going to be nonnegative. Thus in step 6 we check this condition and solve the underlying TSP only if the reduced cost could be negative.

```
for all b do
  for all s do
    Sort q_i^{b,s} and obtain Q^{b,s}.
    Z = (0, 0, ..., 0) ∈ {0,1}^{Q^{b,s}}.
    while Z ≠ (1, 0, ..., 0) do
      if min_k l_Z^{b,s,k} < 0 then
        Compute TSP_Z^b.
        for all k do
          if resource constraint (1) is satisfied for b, s, k and r_Z^{b,s,k} < 0 then
            Select the column corresponding to Z, b, s, k.
          end if
        end for
      end if
      if at least one column has been selected in the above for loop then
        Set Z to be the next lexicographically larger vector.
      else
        Set Z to be the maximum vector Z̄ smaller than Z with |Z̄| ≤ |Z|.
      end if
    end while
  end for
end for
```

Algorithm 1: The pricing algorithm

A possible improvement of the algorithm hinges on rewriting the reduced cost as

$$r_j^{b,s,k} = \sum_{i \in R_j} ((p_1')_i - v_i) + \sum_{i \in R_j} (p_2')^{b,s} \cdot \tilde{\mathbf{d}}^s \cdot t_i + (p_3')^{b,s,k} + (p_2')^{b,s} \cdot \mathbf{d}^s + \mathrm{TSP}_j^b \cdot (p_2')^{b,s} \cdot \bar{\mathbf{d}}^s.$$

and defining $q_i = (p_1')_i - v_i$. Since the remaining terms are nonnegative, it still holds that any site i with $q_i \geq 0$ will not be part of the subset with the minimum reduced cost. We can now assume without loss of generality that $q_1 \leq q_2 \leq \ldots q_Q < 0$ for $Q \leq n_E$ and $q_i \geq 0$ for all other sites i. An advantage in this case is that this order no longer depends on s and b. The adjusted algorithm is presented in Algorithm 2.

Algorithm 1 scans the lexicographical order for each b and s, while Algorithm 2 performs a scan only for each b. From this perspective it seems that the second algorithm might be more efficient. However, from $q_i \leq q_i^{b,s}$, it follows that $Q \geq Q^{b,s}$, and thus there are fewer coordinates to consider in Z in Algorithm 1. We conclude that it is not clear a-priori which of the two algorithms is more efficient.

3.1.3 Branching

The main strategy is to follow a single branch promising to lead to a high quality solution, without exploring the entire branch-and-bound tree. The heuristic procedure is to dive without backtracking. Each LP relaxation on the chosen branch is solved by column generation. The procedure stops when either an integer solution is found or the LP relaxation becomes infeasible.

```
Sort q_i and obtain Q.
Z = (0, 0, ..., 0) ∈ {0,1}^Q.
while Z ≠ (1, 0, ..., 0) do
  for all b do
    Compute TSP_Z^b.
    for all s do
      for all k do
        if resource constraint (1) is satisfied for b, s, k and r_Z^{b,s,k} < 0 then
          Select the column corresponding to Z, b, s, k.
        end if
      end for
    end for
    if at least one column has been selected in the above loop then
      Set Z to be the next lexicographically larger vector.
    else
      Set Z to be the maximum vector Z̄ smaller than Z with |Z̄| ≤ |Z|.
    end if
  end for
end while
```

Algorithm 2: The modified pricing algorithm

At the end the resulting IP is solved using all of the generated columns as a post-processing step.

The mechanisms used in our branching are rounding heuristics and ideas from local branching (Fischetti and Lodi (2003)). To obtain an integer solution, fixing a fractional value to an integer value is commonly used, however, to allow more flexibility, we may not want to directly fix a single variable, but instead to add to the LP relaxation a linear soft-fixing constraint, which by definition must cut off the current fractional solution while allowing flexibility such as imposing that a single variable in a set is 1.

In our case we have three decisions, even though there are only two types of variables. Decisions pertaining to $\mathbf{y}$ are of higher value and importance, and thus these variables should be fixed first. The next decision is which sites to visit. There are no explicit variables reflecting this, however, we can branch on constraints (3). Once all strategies, bases, and sites to visit are selected, it remains to find the actual routes, i.e., fixing $\mathbf{x}$ variables. The framework of branching consists of three general rules:

Rule 1. branching on a base and its strategy (i.e., $y^{b,s}$ variables),

Rule 2. branching on the exploration sites (i.e., the left-hand side of constraints (3)),

Rule 3. branching on routes (i.e., $x_j^{b,s,k}$ variables).

Rule 1 is based on a standard single variable strategy. First, all variables at 1 in the incumbent LP relaxation are fixed to 1. Next the value of the largest fractional variable is found. If this value is larger than 0.5, the underlying variable is fixed to 1 and we dive. Otherwise, we fix the variable with the smallest fractional value to 0 and dive. Each time before fixing a variable to 1, we check if budget constraint (7) is satisfied. If it is, we fix the variable, otherwise we skip it.

To implement Rule 2, we temporarily introduce

$$\sum_{b\in\mathbf{B}}\sum_{s\in\mathbf{S}}\sum_{k\in\mathbf{T}^s}\sum_{j\in J^{b,s,k}}\mathbf{A}_j\cdot x_j^{b,s,k}=\mathbf{e}_{n_1},\quad \mathbf{0}_{n_1}\le\mathbf{e}_{n_1}\le\mathbf{1}_{n_1}\ .$$

For site i in an integer solution, e_i is binary, while it might be fractional in the LP relaxation. Algorithm 3 describes the underlying approach for branching on e's, where $\bar{\mathbf{e}}$ denotes the values in the incumbent LP relaxation. As we go deeper and deeper in the tree, the likelihood of infeasibility increases. For this reason, in step 8 we employ soft variable fixing to increase flexibility.

The same procedure is applied in Rule 3 of branching with respect to route variables $\mathbf{x}$. It can be seen that for Rule 3, Algorithm 3 may result in an infeasible LP since it is possible that not all of the sites required to be visited by Rule 2 can be reachable under constraints (4) and (5), in which case we stop diving deeper along the branch and turn into the post-processing step.

1: Fix all of the variables with value 1 to 1.
2: Let $\bar{B}$ denote the set of all unfixed variables.
3: Let $\Delta=\sum_{i\in\bar{B}}\bar{e}_i$.
4: **if** (Δ is integer) or ($\Delta-\lfloor\Delta\rfloor)<0.5$ and $\Delta>1$) **then**
5: Fix a single variable in $\bar{B}$ using the single variable strategy.
6: **end if**
7: **if** $\Delta-\lfloor\Delta\rfloor\ge 0.5$ **then**
8: Add constraint $\sum_{i\in\bar{B}}e_i\ge\lceil\Delta\rceil$.
9: **if** the resulting LP is infeasible **then**
10: Delete the newly added constraint.
11: Fix a single variable in $\bar{B}$ using the single variable strategy.
12: **end if**
13: **else**
14: Fix e_i with the smallest fractional value $\bar{e}_i$ to 0.
15: **end if**

Algorithm 3: Branching strategy for Rule 2 and 3

3.1.4 Post-processing Step

To obtain the final integer solution and potentially improve the solution quality, we solve the IP with all of the columns generated, as a post-processing step. The optimality gap

$$G_{opt}=\frac{R^*_{LP}-R^*_{IP}}{R^*_{IP}},\tag{19}$$

where R^*_{IP} is the value of the solution obtained and R^*_{LP} is the value of the root node LP relaxation, measures the relative quality of the solution.

3.2 Three-Phase Method

Consider the case when decision vector $\mathbf{y}$ for the GLRPP is fixed. In this case, bases and associated mission strategies are selected by $\mathbf{y}$, and the GLRPP becomes the following problem. The problem

is to decide routes and routing tactics corresponding to each selected base to maximize the profit subject to the constraints on resource consumption for routes and missions. We refer to this problem as the *multi-depot vehicle routing problem with profits* (MDVRPP). This is still not a classical problem due to the existence of routing tactics, multiple depots, and profits on sites. The MDVRPP is easier to solve than the GLRPP because the only decision vector is $\mathbf{x}$.

The three-phase method is composed of the *divide phase*, which is basically a preprocessing step, the *conquer phase*, and the *synthesize phase*. In the *divide phase*, the entire node set is divided into multiple groups, each of which is referred to as a *cluster*. A cluster can have several bases. For each cluster we consider base/mission strategy combinations, and each combination is referred to as a *cluster strategy*. Cluster strategies are enumerated if possible or selected randomly based on assigned scores. In the *conquer phase*, we solve the MDVRPP for each cluster and for each cluster strategy included in the cluster (given $\mathbf{y}$). The MDVRPP is solved by price-and-branch, which is very similar to the one used for the GLRPP. In price-and-branch, the root node LP relaxation is solved by column generation and then the IP is solved with the columns generated at the root node (no branching is employed except by the IP solver). Finally, the *synthesize phase* collects the MDVRPP results for all cluster strategies and solves an auxiliary IP that selects one cluster strategy for each cluster, maximizing the total profit sum for the campaign subject to the budget constraint. The entire algorithm is presented in Algorithm 4. Individual steps are discussed next.

```
Construct the clusters {m̄ is a parameter}
loop
  for all clusters l do
    if the number of bases in cluster l is small enough (5 or less) then
      for all base/strategy pairs do
        Solve the corresponding MDVRPP.
      end for
    else
      for g = 1 to m̄ do
        Generate random budget b_l.
        Generate a random cluster strategy for cluster l.
        Solve the corresponding MDVRPP.
      end for
    end if
  end for
  Solve the synthesis integer program. If a better solution is obtained, we store it.
  if gap less than acceptable tolerance or run time limit is reached then
    Exit.
  end if
end loop
```

Algorithm 4: The outline of the three-phase algorithm

3.2.1 Phase I - Divide

The objective of the *divide phase* is to partition the node set $\mathcal{N}$ into clusters such that the solution space is restricted to include only the routes that visit nodes belonging to the same cluster. We start the *divide phase* with identification of sites that can be visited from each base.

Definition 1 (Reachable)**.** *For base $b \in \mathbf{B}$ and site $i \in \mathbf{E}$, i **is reachable from** b if there exist mission strategy $s \in \mathbf{S}$ and routing tactic $k \in \mathbf{T}^s$ such that the direct round trip between b and i is feasible for routing tactic k. Formally, i is reachable from b if there exist $s \in \mathbf{S}, k \in \mathbf{T}^s$ such that $\mathbf{c}^{s,k} + (c_{bi} + c_{ib}) \cdot \bar{\mathbf{c}}^{s,k} + t_i \cdot \tilde{\mathbf{c}}^{s,k} \leq \mathbf{u}^{s,k}$.*

Note that a base from which no site is reachable or a site which is not reachable from any base is not relevant to the problem and can be deleted from the problem instance. In the remainder of the paper, it is assumed that every considered base has at least one site reachable from the base and every potential site is reachable from at least one base.

Definition 2 (Proximity Set)**.** *For each base $b \in \mathbf{B}$, the proximity set $\mathbf{P}_b$ is defined as the union of $\{b\}$ and the set of sites reachable from b.*

The *divide phase* generates a partition of the whole node set $\mathcal{N}$. This partition is referred to as *clustering* and a member of the partition is referred to as a *cluster*. Collection $\Gamma = \{\mathbf{C}_1, \cdots, \mathbf{C}_l, \cdots, \mathbf{C}_{n_C}\}$ is a clustering of $\mathcal{N}$ and $\mathbf{C}_l$ for $l = 1, \ldots, n_C$ is a cluster of $\mathcal{N}$ if Γ is a partition of $\mathcal{N}$, each proximity set is completely included in a cluster, and all proximity sets within a cluster have a nonempty intersection. Cluster base set $\mathbf{B}_l$ and cluster site set $\mathbf{E}_l$ are defined as $\mathbf{B}_l = \mathbf{C}_l \cap \mathbf{B}, \mathbf{E}_l = \mathbf{C}_l \cap \mathbf{E}$, respectively.

To find a clustering, consider an auxiliary graph whose nodes correspond to proximity sets, i.e., there are n_B nodes. There is an edge between two nodes in the auxiliary graph if and only if the corresponding proximity sets have a non-empty intersection. It is now easy to see that clusters correspond to connected components in this auxiliary graph.

To summarize, in the divide phase we find a clustering by detecting connected components in the auxiliary graph. This is step 1 in Algorithm 4.

3.2.2 Phase II - Conquer

The goal of the *conquer phase* is to first identify a set of bases and strategy combinations for every cluster. This is equivalent to selecting a subset of possible $\mathbf{y}$ vectors. Then, the MDVRPP is solved for each identified mission strategy and base combination (cluster strategy). Both a solution and the upper bound of the maximum profit are recorded to be used in the next phase.

The key step in this phase is the selection of cluster strategies. If the number of bases in the cluster is small enough, all base/strategy pairs are enumerated. However, if enumerating all cluster strategies is too time-consuming, they are selected randomly based on scores, which are greedy estimates. Heuristics with such a feature are known as GRASP. The procedure is performed in two steps. In the first step a random proportion of budget M is assigned to each cluster based on the profit it generates. In the second step bases and strategies are greedily

selected with randomness until the allocated budget is exceeded. Let $\bar{p}_l = \sum_{i \in \mathbf{E}_l} v_i$ be the maximum attainable profit of cluster l.

Intuitively, the more site profit can be generated from a cluster, the larger portion of the budget should be assigned to such a cluster. Thus we define $p_l = \bar{p}_l / \sum_{i \in \mathbf{E}} v_i$ to be the proportion of the total profit attributed to cluster l. Let ϵ be a predetermined parameter. For each cluster l we randomly select a value from $[(1-\epsilon)p_l M, (1+\epsilon)p_l M]$ based on the uniform distribution. The generated value is denoted by b_l and it represents the targeted budget for cluster l (step 10 in Algorithm 4). Note that although the sum of the allocated budget may be bigger than M, it is unlikely that the synthesis phase later would become infeasible because usually some cluster strategies do not exhaust the allocated budget.

We now describe how to generate bases and the corresponding strategies within a cluster. Consider a base b and strategy s. Based on the objective function (2), it is desirable to select a base that can potentially yield substantial profit. To this end, we consider the corresponding proximity set and sum the profit of the sites included in it, $\sum_{i \in \mathbf{P}_b} v_i$. On the other hand, based on (7) it is desirable to select a strategy s with low C^s. We combine these two quantities in

$$\text{score}^{b,s} = \frac{\sum_{i \in \mathbf{P}_b} v_i}{C^s}.$$

We would like to select bases and strategies with high scores.

Consider a cluster l. Let

$$prob_l^{b,s} = \frac{\text{score}^{b,s}}{\sum_{\bar{b} \in \mathbf{B}_l, \bar{s} \in \mathbf{S}} \text{score}^{\bar{b},\bar{s}}}$$

be the probability of selecting base b and strategy s from cluster l. Based on this probability we randomly generate base/strategy pairs. We stop if the summation of C^s of selected pairs exceeds b_l (the last selected base/strategy pair can either be deleted or retained, each with 50% probability). Every time a base is selected, we recalculate the probabilities to reflect the fact that a base can be selected at most once. This corresponds to step 11 in Algorithm 4.

This procedure is repeated $(|\mathbf{S}|+1)^{|\mathbf{B_l}|}$ times, if the condition in step 4 holds, or $\bar{m}$ times, if the condition in step 4 does not hold for each cluster. At the end of each iteration, each cluster has a cluster strategy. A cluster strategy encodes the corresponding $\mathbf{y}$ vector within the cluster. In the next step, for each cluster we solve the underlying MDVRPP by price-and-branch.

For the generated cluster strategy g of cluster l, we record the obtained feasible profit sum $(V_{IP}^*)^{l,g}$, and upper bound $(V_{LP}^*)^{l,g}$ of optimal profit obtained through the root node LP relaxation of the MDVRPP.

At the end, for each cluster l we obtain the set of m_l values $\{(V_{IP}^*)^{l,g}\}_{g=1}^{m_l}$, with $m_l = (|\mathbf{S}|+1)^{|\mathbf{B}_l|}$ for the clusters that can be enumerated and $m_l = \bar{m}$ for the clusters that are too big to enumerate. In the *synthesize phase* all of these values are combined to obtain a feasible solution.

3.2.3 Phase III - Synthesize

The *synthesize phase* solves the problem of assigning one cluster strategy to each cluster to maximize the total profit sum for the entire problem while ensuring that the selected set of strategies satisfies the budget constraint. This is accomplished by solving the following IP:

$$\max_{z^{l,g}} \sum_{l=1}^{n_C} \sum_{g=1}^{m_l} (V_{IP}^*)^{l,g} \cdot z^{l,g} \tag{20}$$

$$\sum_{g=1}^{m_l} z^{l,g} = 1 \qquad l \in \{1, \ldots, n_C\} \tag{21}$$

$$\sum_{l=1}^{n_C} \sum_{g=1}^{m_l} c^{l,g} \cdot z^{l,g} \leq M \tag{22}$$

$$z^{l,g} \in \{0,1\} \qquad l \in \{1, \ldots, n_C\}, g \in \{1, \ldots, m_l\}\,, \tag{23}$$

where $c^{l,g}$ is the sum of the total cost used by cluster strategy g in cluster l. Value $c^{l,g}$ is readily available from the divide phase. Variable $z^{l,g}$ is 1 if cluster strategy g is selected for cluster l, and 0 otherwise. Constraints (21) require that exactly one cluster strategy is assigned to each cluster l. Constraint (22) imposes the budget limit.

When the cluster strategies can be enumerated for all clusters, we run the outer "loop" only once. Let K_{IP}^{*h} be the optimal value of this IP at iteration h of the loop in steps 2-20 of Algorithm 4. The optimal value returned by the algorithm is $\max_h K_{IP}^{*h}$.

When the cluster strategies can be enumerated for all clusters and the objective coefficients in (20) are replaced by $(V_{LP}^*)^{l,g}$, we denote the resulting objective value by K_{LP}^*. Note that in this case there is a single iteration in Algorithm 4. The following can be easily established.

Proposition 1. *If all cluster strategies are enumerated for all clusters, then $R_{opt}^* \leq K_{LP}^* \leq R_{LP}^*$, where R_{opt}^* is the optimal value for GLRPP. Thus the gap is established by*

$$\bar{G}_{opt} = \frac{K_{LP}^* - K_{IP}^{*1}}{K_{IP}^{*1}}\,. \tag{24}$$

If cluster strategies are not enumerated for each cluster, then we still have $K_{LP}^{*h} \leq R_{LP}^*$, where K_{LP}^{*h} is the value obtained by replacing the coefficients in (20) in iteration h by $(V_{LP}^*)^{l,g}$. Unfortunately, it can happen that $K_{LP}^{*h} < R_{opt}^*$ and thus $\min_h K_{LP}^{*h}$ cannot be used in gap computations. In this case the gap is defined by

$$G_{opt} = \frac{R_{LP}^* - \max_h K_{IP}^{*h}}{\max_h K_{IP}^{*h}}\,. \tag{25}$$

4 Numerical Experiments

Numerical experiments for solving GLRPP instances using the two solution methodologies introduced earlier are carried out. In addition, we also evaluate the standard price-and-branch

algorithm, which applies column generation at the root node and then solves the IP over all columns generated. Two metrics are used to evaluate and compare the performance of the solution methodologies. The first metric is the computation time (T_{cal}) representing the efficiency of a solution method and the second metric is the optimality gap (G_{opt},$\bar{G}_{opt}$) representing the effectiveness of a solution method. We consider two types of instances: (1) randomly generated instances, and (2) an instance based on a NASA study.[1] The former are required due to the limited number of NASA based instances.

We first discuss the instance generation procedure. Potential bases are randomly generated inside a circular region from a uniform spatial distribution. The sites are uniformly distributed in the reachable area from a base. The distance is defined as the shortest path distance in the network and thus the constructed distance matrix satisfies the triangular inequality. The number of bases n_B and sites n_E are also randomly selected from a uniform distribution. The profits are selected uniformly at random from the interval $[1, 10]$, which denote the scientific value of the sites. The time (in hours) required to obtain profits at each site is uniformly distributed from the interval $[16, 32]$. Based on the assumption that it costs more to deliver a larger amount of mass to the planetary surface, the mass delivered on planetary surface (MT) is used as a proxy metric for the cost associated with a strategy. We introduce a parameter referred to as the budget tightness (f_B), which is defined as the ratio of the campaign budget M to the maximum possible budget $n_B \cdot \max_{s\in\mathbf{S}} C^s$. The campaign budget for each problem instance is expressed as follows:

$$M = f_B \cdot n_B \cdot \max_{s\in\mathbf{S}} C^s. \tag{26}$$

Tables 1 and 2 summarize the characteristics of the generated GLRPP instances. These instances closely reflect the setting encountered in space exploration (Greeley and Thomas (1995)). The coefficients in Table 1 are determined from the actual study for Mars rover design whose results are available in Ahn et al. (2008) and Hong (2007).

All experiments were carried out on an Intel Xeon 2.79 GHz CPU with 15GB of RAM under Windows Server 2003 operating system. All solution methodologies are implemented in C (Visual Studio 2008 IDE). The CPLEX 12.3 callable library is used to solve the LPs and IPs that are encountered during the solution procedure.

4.1 Estimating Performance Metrics

Basic statistics of the numerical experiments are summarized in Table 3. Among the 100 randomly generated instances, for 47 of them we were able to enumerate all strategies in the three-phase algorithm (we regard 5 or fewer bases in a cluster to be enumerable) and thus gap expression (24) applies. For the remaining 53 instances gap expression (19) has to be used. Since $K^{*h}_{LP} \leq R^*_{LP}$, we call the former instances "Tighter Bound" instances and the remaining instances "General Bound" instances. The mean values indicate that price-and-branch is the best algorithm in terms of the computation time. For the three-phase method, the instances yield only a few proximity

[1]Test instances and results are available at `http://www.4shared.com/zip/q3SmK3pP/GLRPP_instances.html`

Table 1: Characteristics of Instances for Numerical Experiments

	Standard Strategy		**Orbiting Depot Strategy**	
Collective Constraint				
Constraining Resource	Exploration Time		Exploration Time	
Coefficients	$d^1 = 0$	$[h]$	$d^2 = 0$	$[h]$
	$\bar{d}^1 = 0.2$	$[h/km]$	$\bar{d}^2 = 0.2$	$[h/km]$
	$\tilde{d}^1 = 3$	$[-]$	$\tilde{d}^2 = 3$	$[-]$
Consumption Limit	$\bar{l}^1 = 2{,}100$	$[h]$	$\bar{l}^2 = 2{,}100$	$[h]$
Routing Tactics				
Tactic 1	Standard		Standard	
Single-Route Constraint				
Resource	Fuel		Fuel	
Coefficients	$c^{1,1} = 0$	$[kg]$	$c^{2,1} = 0$	$[kg]$
	$\bar{c}^{1,1} = 1.1$	$[kg/km]$	$\bar{c}^{2,1} = 1.1$	$[kg/km]$
	$\tilde{c}^{1,1} = 5.4$	$[kg/h]$	$\tilde{c}^{2,1} = 5.4$	$[kg/h]$
Limit	$l^{1,1} = 700$	$[kg]$	$l^{2,1} = 700$	$[kg]$
Max. Number of Routes	$n^{1,1} = \infty$	$[-]$	$n^{2,1} = \infty$	$[-]$
Tactic 2			Depot-Assisted	
Single-Route Constraint				
Resource			Fuel	
Coefficients			$c^{2,2} = 0$	$[kg]$
			$\bar{c}^{2,2} = 1.1$	$[kg/km]$
			$\tilde{c}^{2,2} = 5.4$	$[kg/h]$
Limit			$l^{2,2} = 1{,}330$	$[kg]$
Max. Number of Routes			$n^{2,2} = 4$	$[-]$
Cost	$C^1 = 15$	$[MT]$	$C^2 = 17$	$[MT]$
Campaign Budget	$f_B \cdot n_B \cdot \max_{s \in \mathbf{S}} C^s$ $[MT]$			

sets within clusters, but the number of clusters vary substantially. Under such circumstances, the underlying MDVRPP problems have to be solved many times by column generation. Even though each MDVRPP does not include many sites, it still requires moderate computational time. For instances with many clusters the overall running time of the three-phase heuristic increases substantially. This is the main cause of the relatively high standard deviation in time. For the branch-and-price method, since we only follow one branch without backtracking, the optimal solution is not necessarily obtained. After fixing a variable, it may happen that either all variables become integral or additional fractional variables appear, thus the computational time may vary significantly (there are other factors which we analyze later). With respect to the optimality gap, the improvement of branch-and-price over price-and-branch is not significant, since on average the number of new columns generated is small and the later added columns tend to have a diminished effect on the objective. However, branch-and-price method should always be no worse than price-and-branch method since the columns generated by the latter are included

Table 2: Instance Generation for Numerical Experiments

Potential Bases and Sites	
Radius of the Region (R)	3410 [km]
Number of Potential Bases (n_B)	$\sim U(50, 150)$
Number of Potential Sites (n_E)	$\sim U(500, 1500)$
Aerial Distribution of Locations	Uniform over all region
Budget Constraint	
Budget Tightness Parameter (f_B)	$2^u/2^5 \quad (u \sim U(0,5))$
Number of Experiments	
No. of Instances	100

in the columns generated by the former.

Table 4 shows that in general branch-and-price performs better than the three-phase method. The second column shows the average improvement in the objective value, which is calculated as $(R^*_{IP} - \max_h K^{*h}_{IP})/\max_h K^{*h}_{IP}$. There are 25 instances where the three-phase method outperforms branch-and-price by an average of 0.99%. On the other hand, for 37 instances branch-and-price is the winner with an average improvement of 0.70%. For the remaining 38 instances, the two algorithms produce the same objective value. The third column is the number of instances where cluster strategies can be enumerated for each cluster under the three-phase method. For enumerable instances, the three-phase method tends to perform better since within each cluster, all base/strategy combinations are enumerated. For non-enumerable instances, branch-and-price tends to outperform.

Estimators for the computational time and optimality gap using the three methodologies are proposed next.

4.1.1 Computation Time

Estimators for the price-and-branch ($\hat{T}_{cal,p}$), branch-and-price ($\hat{T}_{cal,b}$) and three-phase ($\hat{T}_{cal,t}$) computational time ($\hat{T}_{cal,t}$) are proposed as follows:

$$\ln \hat{T}_{cal,p} = K_p + \alpha_p \cdot n_B f_B + \beta_p \cdot n_E + \gamma_p \cdot n_{c,\max}, \tag{27}$$

$$\ln \hat{T}_{cal,b} = K_b + \alpha_b \cdot n_B f_B + \beta_b \cdot n_E + \gamma_b \cdot n_{c,\max}, \tag{28}$$

$$\ln \hat{T}_{cal,t} = K_t + \alpha_t \cdot n_{B,max} + \beta_t \cdot n_{E,max} + \gamma_t \cdot n_{c,\max}, \tag{29}$$

where $n_{B,max}$, $n_{E,max}$ are the maximum numbers of bases, sites in proximity sets comprising a cluster, respectively. We next describe $n_{c,\max}$ (see (30)).

Let us denote $h_i^{s,k} = \mathbf{c}^{s,k} + (c_{bi} + c_{ib}) \cdot \bar{\mathbf{c}}^{s,k} + t_i \cdot \tilde{\mathbf{c}}^{s,k}$. It is very likely that the computational complexity of an instance depends on the number of feasible routes within proximity sets. Given strategy s and routing tactic k we would like to approximate the number of routes using less than or equal to $\mathbf{u}^{s,k}$ units of the resource. The resources consumed by the direct route from base b to

Table 3: Basic Statistics of the Numerical Results

Computation Time for All Instances (T_{cal} [sec])				
	mean	std	min	max
Price-and-Branch	9	14	0	113
Branch-and-Price	376	665	2	3,343
Three-Phase	90	120	1	974
Optimality Gap for General Bound Instances (G_{opt} [%])				
	mean	std	min	max
Price-and-Branch	28.18	28.90	0.13	117.54
Branch-and-Price	27.02	27.68	0.13	116.04
Three-Phase	27.27	27.27	0.16	119.06
Optimality Gap for Tighter Bound Instances ($\bar{G}_{opt}$ [%])				
	mean	std	min	max
Price-and-Branch	1.34	1.81	0	9.39
Branch-and-Price	0.85	1.27	0	8.82
Three-Phase	0.55	0.43	0	1.68
Number of Routes (Columns) Generated				
	mean	std	min	max
Price-and-Branch	1,883	1,038	388	5,266
Branch-and-Price	1,990	1,104	395	5,411

Table 4: Comparing Branch-and-Price with Three-Phase for all 100 Instances

	Instances	Imp[%]	Enumerable	Non-enumerable
Three-phase outperforms	25	-0.99	15	10
Branch-and-price outperforms	37	0.70	6	31
Total	100	0.01	47	53

site i equal to $h_i^{s,k}$. If this value equals $\mathbf{u}^{s,k}$, then only direct routes are possible. On the other hand, if $h_i^{s,k}$ is much smaller than $\mathbf{u}^{s,k}$, then many other routes are possible. As a result we use the sum of the ratios $\frac{\mathbf{u}^{s,k}}{h_i^{s,k}}$ over all sites i in a proximity set as an approximation to the complexity of the proximity set. Parameter $n_{c,\max}$ called the *geometric complexity parameter* is defined as

$$n_{c,\max} = \max_{b \in \mathbf{B}} \sum_{i \in \mathbf{P}_b,\ i \neq b} \max_{s \in \mathbf{S}, k \in \mathbf{T}^s} \frac{\mathbf{u}^{s,k}}{h_i^{s,k}} \,. \tag{30}$$

Note that for each $i \in \mathbf{P}_b$, by definition of $\mathbf{P}_b$, we have $\max_{s \in \mathbf{S}, k \in \mathbf{T}^s} \frac{\mathbf{u}^{s,k}}{h_i^{s,k}} \geq 1$.

Parameters for the estimations are identified from the results of the numerical experiments using the least-squares method. Identified parameters and R^2_{adj} values (adjusted coefficients of multiple determination) of the estimators are presented in Table 5. For branch-and-price and

price-and-branch methodologies, we choose $n_B \cdot f_B, n_E$, and $n_{c,max}$ as predictors. When $n_B \cdot f_B$ is large, according to (26), more $\mathbf{y}$ variables are likely to have value one in the root node LP relaxation, which influences the size of the branch-and-bound tree. Large n_E and $n_{c,max}$ tend to increase the number of feasible routes and thus the running time. For the three-phase method, since the whole graph is partitioned into clusters and the budget is allocated to each cluster, we choose $n_{B,max}, n_{E,max}, n_{c,max}$ as predictors. They tend to increase the number of feasible routes in a cluster and thus the running time.

Table 5: Parameters for the Computational Time

Price-and-Branch		Branch-and-Price		Three-Phase	
Parameter	Value	Parameter	Value	Parameter	Value
K_p	-1.525	K_b	0.188	K_t	0.855
α_p	-0.023	α_b	-0.045	α_t	0.148
β_p	0.005	β_b	0.009	β_t	0.019
γ_p	0.102	γ_b	0.121	γ_t	0.006
R_p^2	0.797	R_b^2	0.831	R_t^2	0.618

4.1.2 Optimality Gap

Estimators for the optimality gap under the general bound from the root node LP relaxation with price-and-branch, branch-and-price and three-phase method are proposed considering the observation that the optimality gap increases as the budget tightness factor f_B decreases. With tight budget there are more fractional $\mathbf{y}$ values in the LP relaxation solution, which leads to larger optimality gaps.

It is also expected that the changes in G_{opt} are larger for smaller f_B. Estimator $\hat{G}_{opt,p}$ for price-and-branch, $\hat{G}_{opt,b}$ for branch-and-price, and $\hat{G}_{opt,t}$ for the three-phase method are proposed as follows:

$$\hat{G}_{opt,p} = K_{0,p} + K_{1,p} \cdot f_B + K_{2,p} \cdot \frac{1}{f_B} \tag{31}$$

$$\hat{G}_{opt,b} = K_{0,b} + K_{1,b} \cdot f_B + K_{2,b} \cdot \frac{1}{f_B} \tag{32}$$

$$\hat{G}_{opt,t} = K_{0,t} + K_{1,t} \cdot f_B + K_{2,t} \cdot \frac{1}{f_B} \ . \tag{33}$$

The parameters are identified using the least-squares method and presented in Figure 5, Figure 6, and Figure 7.

4.2 NASA Case

Here we present computational results for a true NASA instance. The parameters given in Table 1 closely reflect the setting for Mars explorations and the budget is set to 700 (Ahn et al. (2008)).

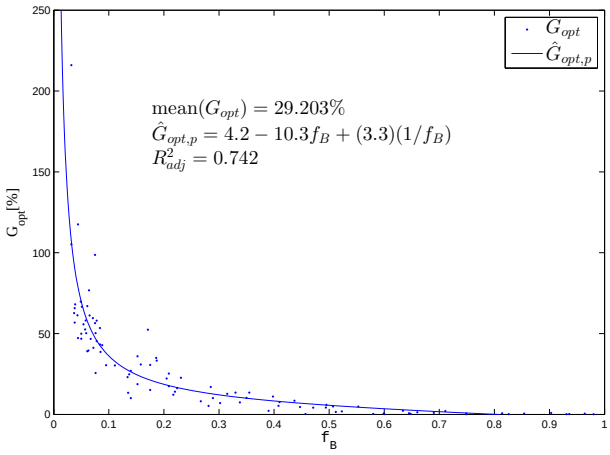

Figure 5: Gap for Price-and-Branch

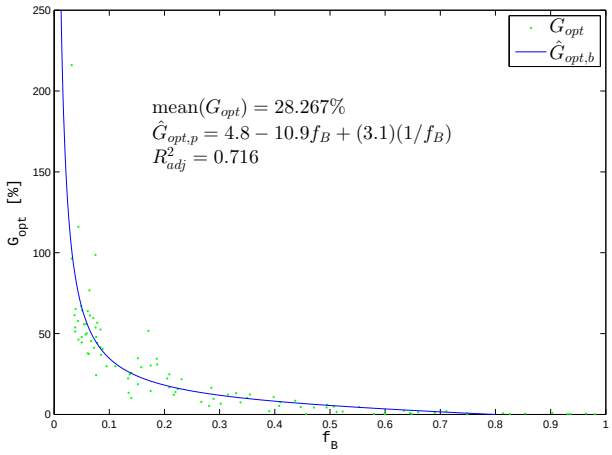

Figure 6: Gap for Branch-and-Price

The longitude and latitude of the 153 exploration sites are obtained from Greeley and Thomas (1995). Bases are also chosen from the 153 sites. The instance generated is non-enumerable for the three-phase method. Since running time is not a major concern, we ran both the branch-and-price and three-phase method, which gave solution values of 127 within 31 seconds and 126 within 55 seconds, respectively.

We report the solution by branch-and-price in Figure 8, which shows the most site-intensive part on Mars. Sites are indexed by numbers. The locations in rectangles are the selected bases and the locations in ovals denote the sites served by the corresponding base. On the entire Mars' surface overall 11 bases are selected of which 10 bases (index 32, 17, 88, 120, 25, 72, 125, 128, 104, 30) adopted the orbiting depot strategy and a single base (index 102) adopted the standard strategy. Totally 62 sites are visited by 54 generated routes, of which 33 routes use the depot-assisted tactic under the orbiting depot strategy, 18 routes use the standard strategy under the orbiting depot strategy, and 3 routes use the standard tactic under the standard strategy.

5 Summary and Future Work

This paper deals with a new routing problem class referred to as the generalized location routing problem with profits. Given locations of potential bases and sites, and profit values assigned to the sites, the GLRPP maximizes the sum of profit obtained over a campaign by making decisions on selection of bases to use, selection of strategies for the missions using the bases, selection of sites and routes to visit the sites, and selection of routing tactics, with constraints imposed on each route, each mission, and the overall campaign. A mathematical formulation for the problem is provided and two solution methodologies to find an approximate solution are proposed. Performance analysis of the two solution methods indicates that if solution quality is the only concern, to choose a proper method, we should first carry out the divide phase of the three-phase method. If all the resulting clusters have a small number of bases such that cluster

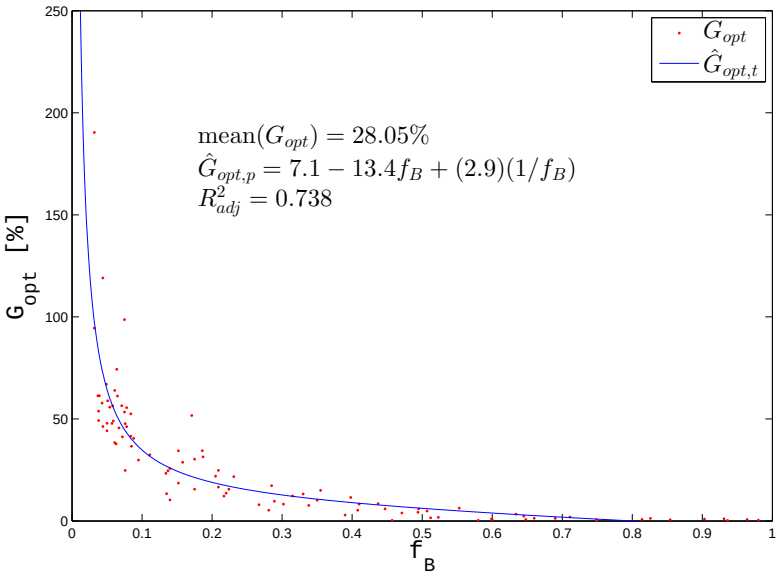

Figure 7: Gap for Three-Phase Method

strategies can be enumerated, then the three-phase method should be applied. Otherwise, the branch-and-price algorithm should be the choice.

Two applications using the GLRPP framework - planetary surface exploration and college football recruiting - have already been studied (Ahn et al. (2008)). Military surface operation and gas extraction problems can be also formulated and solved using this framework, which is suggested as future work. A limitation of the GLRPP framework is the way it deals with "profits." Under the current framework, profit value to visit a site is one dimensional. Sometimes there are multiple stakeholder groups interested in visiting sites and obtaining profits. Valuations of a site from different stakeholder groups will likely be different. It means that the value of visiting a site should be considered as a vector, not a scalar. A revision of the current framework such that it can deal with the profit vector for each site is also suggested as future work. One possible way would be to maximize the profit sum of the least satisfied stakeholder.

References

Ahn, J., O. de Weck, J. Hoffman. 2008. An optimization framework for global planetary surface exploration campaigns. *Journal of the British Interplantery Society* **61** 487–498.

Archetti, C., A. Hertz, M.G. Speranza. 2011. Metaheuristics for the team orienteering problem. *Journal of Heuristics* **13** 49–76.

Balas, E. 1989. The prize collecting traveling salesman problem. *Networks* **19** 621–636.

Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance. 1998. Branch-and-price: Column generation for solving huge integer programs. *Operations Research* **46** 316–329.

Berger, R., C. Coullard, S. Daskin. 2007. Location-routing problems with distance constraints. *Transportation Science* **41** 29–43.

Bérubé, J.F., M. Gendreau, J.Y. Potvin. 2009. An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. *European Journal of Operational Research* **194** 39–50.

Butt, S.E., T.M. Cavalier. 1994. A heuristic for the multiple tour maximum collection problem. *Computer and Operations Research* **21** 101–111.

Butt, S.E., D.M. Ryan. 1999. An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computer and Operations Research* **26** 427–441.

Chao, I., B. Golden, E. Wasil. 1996. The team orienteering problem. *European Journal of Operations Research* **88** 475–489.

Dell'Amico, M., F. Maffioli, P. Värbrand. 1988. On prize-collecting tours and the asymmetric traveling salesman problem. *International Transactions in Operational Research* **31** 515–530.

Desaulniers, G., J. Desrosiers, M.M. Solomon, eds. 2005. *Column Generation*. Springer Verlag.

Duhamel, C., P. Lacomme, C. Prins, C. Prodhon. 2010. A grasp×els approach for the capacitated location-routing problem. *Computers and Operations Research* **37** 1912–1923.

Feillet, D., P. Dejax, M. Gendreau. 2005. Traveling saleman problem with profits. *Transportation Science* **39** 188–205.

Fischetti, M., A. Lodi. 2003. Local branching. *Mathematical Programming Series B* **98** 23–47.

Greeley, R., P. Thomas. 1995. *Mars Landing Site Catalog: The Electronic Version*. http://cmex.ihmc.us/Marstools/Mars_Cat/Mars_Cat.html (Date accessed Aug 27, 2009).

Gueguen, C. 1999. Méthodes de résolution exacte pour les problèmes de tournées de véhicules. Unpublished doctoral dissertation, École Centrale Paris.

Hong, S. 2007. Design of power systems for extensible surface mobility systems on the moon and mars. Master's thesis, Massachusetts Institute of Technology.

Karaoglan, I., F. Altiparmak, I. Kara, B. Dengiz. 2011. A branch and cut algorithm for the location-routing problem with simultaneous pickup and delivery. *European Journal of Operational Research* **211** 318–332.

Kataoka, S., S. Morito. 1988. An algorithm for the single constraint maximum collection problem. *Journal of Operational Research Society Japan* **31** 515–530.

Laporte, G., F. Louveaux, H. Mercure. 1989. Models and exact solutions for a class of stochastic location-routing problems. *European Journal of Operational Research* **39** 71–78.

Laporte, G., S. Martello. 1990. The selective traveling salesman problem. *Discrete Applied Mathematics* **26** 193–207.

Laporte, G., Y. Nobert, S. Taillefer. 1988. Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation Science* **22** 161–172.

Min, H., V. Jayaraman, R. Srivastava. 1998. Combined location-routing problem: A synthesis and future research directions. *European Journal of Operational Research* **108** 1–15.

Nagy, G., S. Salhi. 2007. Location-routing: Issues, models and methods. *European Journal of Operational Research* **177** 649–672.

NASA, other space agencies. 2007. The global exploration strategy: The framework for coordination. Tech. rep., NASA and 13 space agencies from around the world. Available at http://www.nasa.gov/pdf/178109main_ges_framework.pdf.

Souffriau, W., P. Vansteenwegen, G. Vanden Berghe, D. Van Oudheusden. 2011. Multiconstraint team orienteering problem with multiple time windows. *Transportation Science* To appear.

Srivastava, R., W. Benton. 1990. The location-routing problem: considerations in physical distribution system design. *Computers and Operations Research* **17** 427–435.

Tang, H., E. Miller-Hooks. 2005. A tabu search heuristic for the team orienteering problem. *Computer and Operations Research* **32** 1379–1407.

Tsiligirides, T. 1984. Heuristic methods applied to orienteering. *The Journal of the Operational Research Society* **35** 797–809.

Tuzun, D., L. Burke. 1999. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research* **116** 87–99.

Vansteenwegen, P., W. Souffriau, D. Van Oudheusden. 2011. The orienteering problem: A survey. *European Journal of Operational Research* **209** 1–10.

A Nomenclature

$\mathbf{B}$	index set of potential bases
$\mathbf{C}$	distance matrix
$\mathbf{E}$	index set of potential exploration sites
$J^{b,s,k}$	index set of subsets which, combined with base b, make feasible routes with respect to routing tactic k of strategy s
$\mathbf{R}_j$	subset of potential exploration sites on route j
$\mathbf{S}$	index set of mission strategies
$\mathbf{T}^s$	index set of routing tactics for mission strategy s
$x_j^{b,s,k}$	decision variable, which is 1 if route determined by base b and site subset $\mathbf{R}_j$ using routing tactic k and mission strategy s is selected, and 0 otherwise
$y^{b,s}$	decision variable, which is 1 if mission strategy s is selected by base b, and is 0 otherwise
f_j	profit sum of potential exploration sites on route j
t_i	time required to obtain profits at potential exploration site i
v_i	profit value assigned to potential exploration site i
$\mathbf{c}^{s,k}$	single-route constraint per-route resource consumption coefficient vector (tactic k and strategy s)
$\bar{\mathbf{c}}^{s,k}$	single-route constraint on-arc resource consumption coefficient vector (tactic k and strategy s)
$\tilde{\mathbf{c}}^{s,k}$	single-route constraint on-site resource consumption coefficient vector (tactic k and strategy s)
$\mathbf{d}^s$	collective constraint per-route resource consumption coefficient vector (strategy s)
$\bar{\mathbf{d}}^s$	collective constraint on-arc resource consumption coefficient vector (strategy s)
$\tilde{\mathbf{d}}^s$	collective constraint on-site resource consumption coefficient vector (strategy s)

$\mathbf{u}^{s,k}$	single-route constraint resource consumption limit vector (tactic k and strategy s)
$\bar{\mathbf{u}}^{s}$	collective constraint resource consumption limit vector (strategy s)
$\mathbf{n}^{s}$	maximum number of routes vector (strategy s)

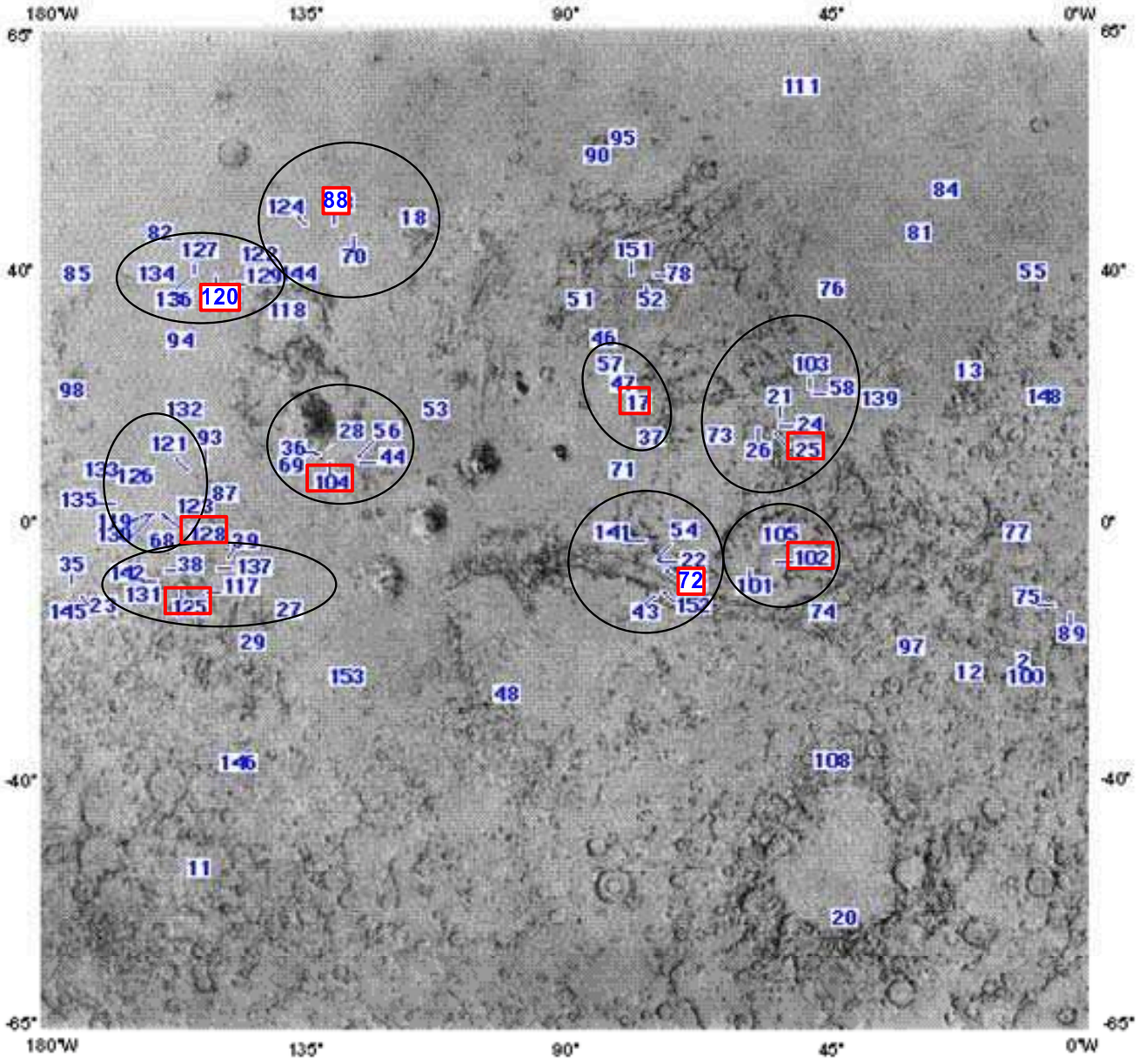

Figure 8: An Illustration of Bases and Sites Selection for Western Sites on Mars