The Multiple Choice Elementary Constrained Shortest Path Problem

Karen Smilowitz • Guangming Zhang

Department of Industrial Engineering and Management Sciences, Northwestern University,

ksmilowitz@northwestern.edu • g-zhang@northwestern.edu

Abstract

This paper considers a variation of the Elementary Constrained Shortest Path Problem in which nodes are separated into subsets and a feasible path through the network may visit at most one node from each subset. We refer to this problem as the Multiple Choice Elementary Constrained Shortest Path Problem (MC-ECSPP). The MC-ECSPP arises as a subproblem in branch-and-price approaches for variations of the vehicle routing problem in which the nodes to be visited are chosen among subsets. We present methods to obtain bounds and feasible solutions for the MC-ECSPP. Further, we incorporate these methods into a branch-and-price approach to solve a variation of the vehicle routing problem.

This paper introduces a variation of the Elementary Constrained Shortest Path Problem (ECSPP), also known as the Elementary Shortest Path Problem with Resource Constraints or Time Windows. The ECSPP finds a path of minimum cost between a source node and a sink node, visiting each node in the network at most once. The path may be constrained by the travel time of the path and time windows on the nodes. We consider a variation of the ECSPP in which nodes are separated into subsets and a feasible path through the network may visit at most one node from each subset. We refer to this problem as the Multiple Choice Elementary Constrained Shortest Path Problem (MC-ECSPP). The MC-ECSPP is defined on a directed network of nodes and arcs, with a cost and a non-negative travel time associated with each arc. The set of nodes is divided into subsets. The MC-ECSPP finds a path with the minimum total cost that visits at most one node from each subset, constrained by the travel time of the path and time windows on nodes.

In this paper, we analyze the MC-ECSPP and develop methods to obtain bounds and feasible solutions, making use of recent advances in solution methods for the ECSPP. As shown in this paper, the MC-ECSPP is important because it arises as a subproblem in a branch-and-price method for a variation of the vehicle routing problem, known as the Multi-Resource Routing Problem (MRRP). Therefore, the second part of the paper incorporates the solution and bounding methods for the MC-ECSPP in a branch-and-price approach for the MRRP. We show that these methods can improve the implementation of a branch-and-price method for the MRRP.

Section 1 discusses the motivation for the MC-ECSPP and reviews work on related shortest path problems. Section 2 presents the formulation of the MC-ECSPP. Section 3 introduces bounding and solution methods for the MC-ECSPP, and Section 4 incorporates these methods within a branch-and-price approach for the MRRP. Section 5 summarizes the paper.

1 Background

The ECSPP arises frequently as a pricing problem in branch-and-price approaches to solve the Vehicle Routing Problem (VRP) and variations of the VRP, including the Pickup and Delivery Problem; see, for example, Dumas *et al.* (1991); Desrochers *et al.* (1992) and Savelsbergh and Sol (1998). When the number of vehicle routes is excessively large, column generation reduces the computational effort required to solve the linear relaxation of the VRP at the nodes of the branch-and-price tree by considering a reduced subset of routes. New routes are added to this subset according to a pricing problem which searches for routes with negative reduced costs. The pricing problem is often solved as an ECSPP, starting and ending at the depot, with costs that are determined by the dual values from the linear relaxation of the master problem.

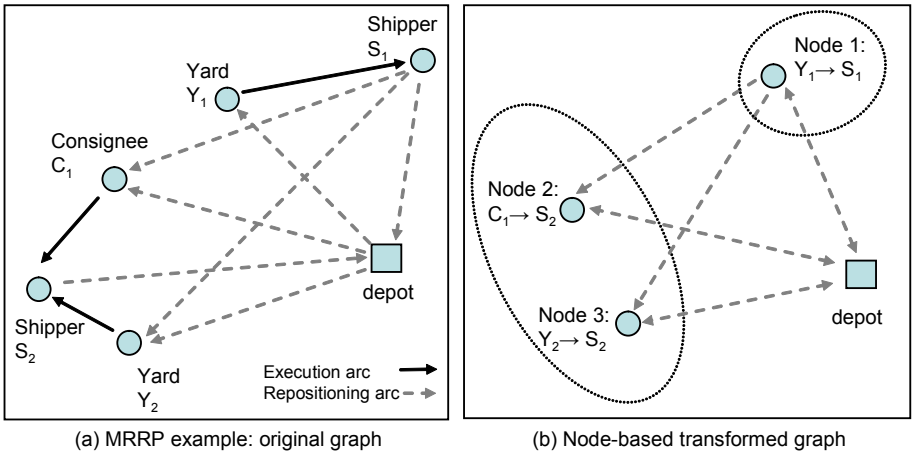


Figure 1: An example of a multi-resource routing problem with flexible tasks

We consider a variation of VRP in which the nodes to be visited are chosen among options within subsets. Such problems occur in drayage operations (i.e., loaded and empty tractor and trailer trips between rail yards, shippers, and consignees). Vehicle routes serve *tasks* that can be satisfied by several possible *executions*. A task is *flexible* if it involves a choice of origin or destination. For example, supplying a shipper with an empty trailer is a flexible task if the origin

of the trailer is chosen among several locations, such as equipment yards or consignees with empty trailers to reposition. A task is *fixed* if only one execution is possible. We refer to this problem as the Multi-Resource Routing Problem (MRRP), see Smilowitz (2006). An example is provided in Figure 1. In Figure 1(a), there are two tasks to be performed by tractors from a depot: a trailer supply from equipment yard Y_1 to shipper S_1 and a trailer supply to shipper S_2 from yard Y_2 or consignee C_1. Dotted lines represent tractor movements between tasks; solid lines represent task executions.

In Figure 1(b), the MRRP graph is transformed to model the problem as an asymmetric VRP. The nodes (representing executions) are partitioned into subsets (representing tasks). For fixed tasks, such as Y_1 to S_1, subsets contain only one node; for flexible tasks, such as the trailer supply to S_2, subsets contain a node for each feasible execution. The VRP solution must visit exactly one node in each subset. The VRP can be solved with a branch-and-price method, with modifications in the pricing problem for flexible tasks. The pricing problem can be modeled as an MC-ECSPP to find the shortest path in terms of cost, beginning and ending at the depot, visiting at most one node from each subset.

1.1 The Multi-Resource Routing Problem

The MRRP designs routes to serve tasks which may have a choice in executions. The tasks are represented by the set $\mathcal{T}$; each task $k \in \mathcal{T}$ is satisfied by an execution chosen from the set $\mathcal{E}_k$. Let $\mathcal{R}$ denote the set of feasible routes, with cost c_r for $r \in \mathcal{R}$. For task $k \in \mathcal{T}$, let δ_{re} equal 1 if execution $e \in \mathcal{E}_k$ is on route $r \in \mathcal{R}$ and 0 otherwise. The decision variable y_r indicates whether or not route $r \in \mathcal{R}$ is chosen. The set partitioning formulation of the MRRP is:

$$\min \sum_{r \in \mathcal{R}} c_r y_r \tag{1a}$$

subject to

$$\sum_{r \in \mathcal{R}} \sum_{e \in \mathcal{E}_k} \delta_{re} y_r \geq 1 \qquad \forall k \in \mathcal{T} \tag{1b}$$

$$y_r \in \{0, 1\} \qquad \forall r \in \mathcal{R} \tag{1c}$$

The objective function (1a) minimizes the cost of all routes chosen. We define c_r to minimize fleet size as the primary objective and travel cost as the secondary objective. Equations (1b) ensure that one execution of each tasks is chosen. With the triangle inequality in route costs, these constraints hold at equality. Equations (1c) define the binary decision variables for each route.

Smilowitz (2006) proposes a branch-and-price algorithm to solve formulation (1). At each node of the branch-and-price tree, column generation is used to solve the linear relaxation of formulation (1) with a restricted route set $\mathcal{R}'$, rather than enumerating all routes in $\mathcal{R}$. The pricing problem to generate candidate routes is solved to optimality for small problem instances with an exact method based on k-cycle elimination from Irnich and Villeneuve (2004) adapted to allow for the multiple choice subsets. Larger instances are solved with a trip insertion heuristic.

1.2 Related shortest path problems

Efficient solution methods for the ECSPP, shown in Dror (1994) to be NP-hard, are critical for implementing branch and price for the VRP. Desrosiers *et al.* (1995) and Irnich and Desaulniers (2004) review solution methods for the ECSPP. One approach is to relax the elementary path constraints and use label correcting or label setting methods for the constrained shortest path problem (CSPP). However, this approach can produce weak lower bounds for cyclic graphs with negative arc costs, see Feillet *et al.* (2004). Irnich and Villeneuve (2004) and Feillet *et al.* (2004) develop efficient methods of eliminating cycles, building on the 2-cycle elimination method by Kolen *et al.* (1987). Feillet *et al.* (2004) show how data requirements increase significantly when using a label-correcting approach to the ECSPP. These difficulties are compounded with the existence of node subsets in the MC-ECSPP. The number of non-dominated paths that must be maintained throughout the labeling algorithm grows prohibitively large in the MC-ECSPP; see Smilowitz (2006).

Constraints on the nodes visited along a path, similar to those in the MC-ECSPP, appear in variations of the CSPP on acyclic graphs. Villeneuve and Desaulniers (2005) introduce the CSPP with forbidden paths, where a set of pre-specified sub-paths are explicitly prohibited from feasible solutions. Although the MC-ECSPP can be formulated as an extension of this problem, the elementary path and multiple-choice constraints become too large to enumerate even for small problem instances. Crainic and Florian (2005) consider a problem similar to the MC-ECSPP on an acyclic graph which finds the shortest path through a "logistics chain" which chooses tasks within node subsets. The solution methods are difficult to apply to the MC-ECSPP when the graph has negative cost cycles.

The MC-ECSPP combines the elementary constraints of the ECSPP and the multiple choice constraints of the CSPP variations discussed above. Algorithms for the ECSPP do not adapt well to the complications of the MC-ECSPP, as shown in Smilowitz (2006). In this paper, we show how this can be resolved with modifications to the MC-ECSPP.

2 The multiple choice elementary constrained shortest path problem

The MC-ECSPP is solved on a directed graph $G = \{N, A\}$ of nodes N and arcs A. Let node $i = 0$ denote the source node and $i = n$ denote the sink node. The node set $N \backslash \{0, n\}$ is partitioned into K subsets, $\{N_1, N_2, ...N_K\}$. Each arc $(i, j) \in A$ has a travel time, t_{ij}, which is nonnegative and a cost, c_{ij}, which is unrestricted in sign. When the MC-ECSPP occurs as a subproblem in branch-and-price methods for the MRRP, arc costs may be negative due to dual values; i.e., π_i for node $i \in N_k$, related to constraints (1b) for task k: $c_{ij} = \pi_i + t_{ij}$. Note that there are no arcs between nodes within the same subset. The duration of an execution represented by node i can be included in the calculation of t_{ij}. The time window for visiting node i is $[a_i, b_i]$. The total length of any path cannot exceed L, which may represent a driver work shift.

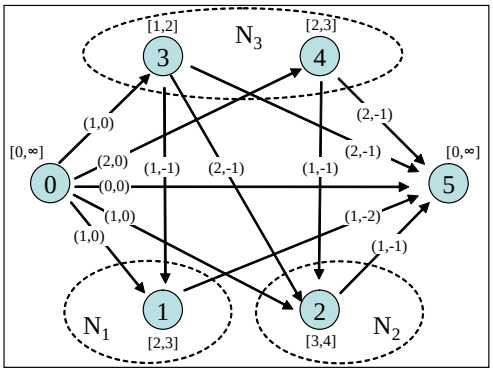


Figure 2: Example of a multiple choice elementary constrained shortest path problem

Figure 2 presents an example of an MC-ECSPP instance. There are three subsets (N_1, N_2, N_3) that can be visited between the source node 0 and the sink node 5. Subsets N_1 and N_2 each consist of a single node, and subset N_3 consists of two nodes (3 and 4). The figure shows the time window for each node, and travel time and cost of each arc. Only feasible arcs are shown, which do not violate time window constraints or path length constraints. Since an arc of zero cost from the source to the sink always exists, there will always be a feasible solution to the MC-ECSPP.

Let $x_{ij} = 1$ if arc $(ij) \in A$ is in the shortest path and 0 otherwise. Let T_i denote the time at which node $i \in N$ is visited on the path. The MC-ECSPP is formulated as follows.

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2a}$$

subject to

$$\sum_{j\in N:(i,j)\in A} x_{ij} - \sum_{j\in N:(j,i)\in A} x_{ji} = \begin{cases} 1 & i = 0 \\ 0 & \forall i \in N \setminus \{0, n\} \\ -1 & i = n \end{cases} \tag{2b}$$

$$\sum_{i\in N_k} \sum_{j\in N:(i,j)\in A} x_{ij} \leq 1 \qquad \forall k = 1..K \tag{2c}$$

$$x_{ij}\left(T_i + t_{ij} - T_j\right) \leq 0 \qquad \forall (i,j) \in A \tag{2d}$$

$$a_i \leq T_i \leq b_i \qquad \forall i \in N \tag{2e}$$

$$T_n - T_0 \leq L \tag{2f}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in A \tag{2g}$$

The objective function (2a) minimizes the path cost. Constraints (2b) force the path to begin at the source node and end at the sink node and maintain flow balance at the other nodes. Constraints (2c) ensure that at most one node in each subset appears on the path. Constraints (2d) and (2e) enforce time windows at the nodes. Constraint (2f) limits the resources consumed by the path. Constraints (2g) define arc variables as binary decisions.

Dror (1994) shows that the ECSPP is NP-Hard. The MC-ECSPP is NP-Hard since an ECSPP instance can be transformed in polynomial time to an MC-ECSPP instance in which each subset consists of a single node. It is critical to reduce problem size when possible. Removing dominated nodes can reduce the size of an MC-ECSPP instance. We define a dominated node as follows.

Definition 1 *Node $i \in N_k$ is dominated by node $j \in N_k$ if*

1. $t_{li} \geq t_{lj}$ *and* $t_{il} \geq t_{jl}$ $\forall l \in N \backslash N_k$
2. $c_{li} \geq c_{lj}$ *and* $c_{il} \geq c_{jl}$ $\forall l \in N \backslash N_k$
3. $a_i \geq a_j$ *and* $b_i \leq b_j$

According to Definition 1, node $i \in N_k$ is dominated by node $j \in N_k$ if node i has (1) equal or longer travel times to and from all nodes not in N_k, (2) equal or higher costs to and from all nodes

not in N_k, and (3) an equal or smaller time window. Node i can be removed from N_k because of node dominance. If, for all subsets, the nodes in the subset are dominated by one node, then each subset can be reduced to a single node, and the MC-ECSPP becomes an ECSPP.

3 Solution method

This section presents solution and bounding methods for the MC-ECSPP. The difficulties in solving the MC-ECSPP arise from the combination of multiple choice and elementary path constraints, see Table 1. Smilowitz (2006) considers both constraints with an exact k-cycle method. Memory requirements for labels limit the use of this method to small instances. We develop a two-phase method to decompose the problem into an aggregated bounding subproblem involving only elementary path constraints and an expansion subproblem involving only multiple choice constraints.

	Multiple choice constraints	Elementary constraints
MC-ECSPP • Adapted K-cycle: Smilowitz (2006)	✓	✓
Two-phase method • Aggregated bounding subproblem - Lower bound: §3.1.1 - Conservative upper bound: § 3.1.2 - One-node upper bound: § 3.1.3		✓
• Expansion subproblem §3.2	✓	

Table 1: Overview of solution and bounding methods for the MC-ECSPP

Section 3.1 presents the aggregated bounding subproblems for lower and upper bounds, in which nodes in each subset are aggregated into a single node and an ECSPP is solved on the aggregated network. Section 3.2 presents the expansion subproblems for feasible solutions to the MC-ECSPP, in which the aggregated solution path is expanded by choosing one original node from each subset along the path. Initially, all node subsets are disjoint. In Section 3.3, the two-phase method is generalized to include non-disjoint node subsets.

3.1 Aggregated bounding subproblem: disjoint subsets

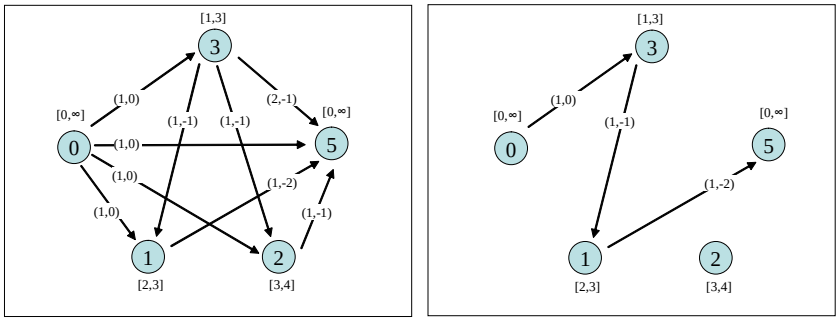
We present aggregated subproblems that result in bounds for the MC-ECSPP. Approaches to determine the properties of the aggregated nodes (cost, travel time, and time windows) are introduced for lower bounds in Section 3.1.1, and for upper bounds in Sections 3.1.2 and 3.1.3.

3.1.1 Lower bounds: disjoint subsets

We obtain lower bounds on the MC-ECSPP solution by aggregating nodes within each subset into a single node with the best (possibly infeasible) combination of subset parameters. The aggregated graph $G^+_{LB} = \{N^+_{LB}, A^+_{LB}\}$ is defined as follows. The node set N^+_{LB} consists of $K + 2$ nodes, with one node representing each subset, and the source and sink nodes. The time windows of the aggregated nodes are defined by the widest limits of the original subset. For $k \in \{1 \dots K\}$, $a^+_k = \min_{i \in N_k} a_i$ and $b^+_k = \max_{i \in N_k} b_i$. The arc set A^+_{LB} consists of all feasible arcs connecting the nodes of N^+_{LB}. The travel time for an arc between the aggregated nodes for subsets k and l is defined as: $t^+_{kl} = \min_{i \in N_k, j \in N_l} t_{ij}$ and the cost is: $c^+_{kl} = \min_{i \in N_k, j \in N_l} c_{ij}$. Arcs to/from the depot are defined in the same manner. An ECSPP is solved on the aggregated graph using a label correcting method that incorporates the k-cycle elimination method of Irnich and Villeneuve (2004).

Theorem 1 *The optimal solution to the aggregated problem as defined above is a lower bound on the shortest path through the original MC-ECSPP network.*

Proof: Let $Z(G)$ denote the cost of the shortest path for the original graph, $G = \{N, A\}$, and let $Z(G^+_{LB})$ denote the cost of the shortest path for the aggregated graph, $G^+_{LB} = \{N^+_{LB}, A^+_{LB}\}$. Let $G' = G \cup G^+_{LB}$, whose optimal solution has a cost $Z(G')$. Since $G \subseteq G'$, then $Z(G) \geq Z(G')$. In G', nodes from G^+_{LB} dominate nodes from G according to Definition 1; thus nodes from G can be eliminated from G' and $Z(G') = Z(G^+_{LB})$. As a result, $Z(G) \geq Z(G^+_{LB})$. □



(a) Lower bound graph (b) Lower bound solution path

Figure 3: Node aggregation for lower bounds

Figure 3 illustrates how a lower bound is created for the original example from Figure 2. We focus on the aggregation of subset N_3, since N_1 and N_2 each contain one node. The time window for the aggregated node is $[1, 3]$, which is large enough to encompass both nodes in N_3. The arc

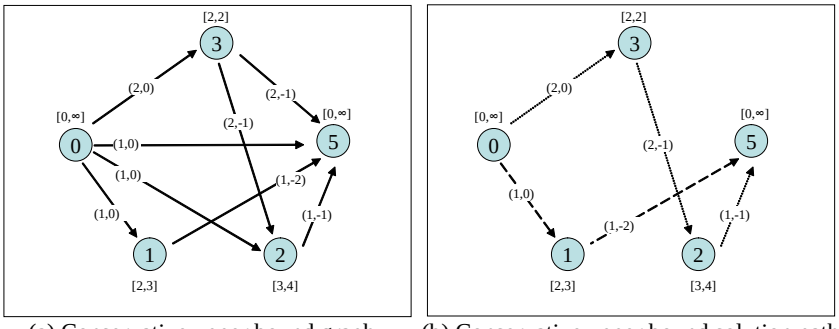
lengths and costs in to and out of the aggregated node for N_3 assume their lowest values. Solving an ECSPP over the aggregated network yields the following solution: $0-3-1-5$ with a length of 3 units and a cost of -3 units, which is a lower bound on the optimal solution to the MC-ECSPP.

3.1.2 Conservative upper bounds: disjoint subsets

While the lower bound aggregation assumes the best parameter values, the conservative upper bound aggregation is obtained by taking the most restrictive values to form $G^+_{UB_c} = \{N^+_{UB_c}, A^+_{UB_c}\}$. The time windows of the aggregated nodes are defined by the narrowest limits. For $k \in \{1 \dots K\}$, $a^+_k = \max_{i \in N_k} a_i$ and $b^+_k = \min_{i \in N_k} b_i$. If $a_k > b_k$, the aggregated node for subset k is removed. Between subsets k and l, $t^+_{kl} = \max_{i \in N_k, j \in N_l} t_{ij}$ and $c^+_{kl} = \max_{i \in N_k, j \in N_l} c_{ij}$. If at least one arc between subsets is infeasible, no arc is created between the aggregated nodes. Arcs to/from the depot are defined in the same manner. An ECSPP is solved on the aggregated graph.

Theorem 2 *The optimal solution to the aggregated problem as defined above produces an upper bound on the shortest path through the original MC-ECSPP network.*

Proof: Let $Z(G)$ denote the cost of the shortest path for the original graph, $G = \{N, A\}$, and let $Z(G^+_{UB_c})$ denote the cost of the shortest path for the aggregated graph, $G^+_{UB_c} = \{N^+_{UB_c}, A^+_{UB_c}\}$. Let $G' = G \cup G^+_{UB_c}$, whose optimal solution has a cost $Z(G')$. Since $G^+_{UB_c} \subseteq G'$, then $Z(G^+_{UB_c}) \geq Z(G')$. In G', nodes from G dominate nodes from $G^+_{UB_c}$ according to Definition 1; thus nodes from $G^+_{UB_c}$ can be eliminated from G' and $Z(G') = Z(G)$; thus, $Z(G^+_{UB_c}) \geq Z(G)$. □



(a) Conservative upper bound graph (b) Conservative upper bound solution paths

Figure 4: Node aggregation for upper bounds

In Figure 4, the MC-ECSPP graph from Figure 2 is aggregated to obtain an upper bound. The time window for the aggregated node for subset N_3 is $[2, 2]$. The arc lengths and costs in to

and out of the aggregated node for N_3 assume their highest values. There are two optimal paths: $0-3-2-5$ and $0-1-5$, each with a cost of -2 units.

Some connections between subsets may become infeasible as a result of the conservative aggregation. In the original MC-ECSPP graph in Figure 2, it is possible to move from subset N_3 to subset N_1 via arc $(1,3)$, yet it is not possible to do so in the aggregated graph since arc $(2,3)$ in the original graph is not feasible. As a result, the upper bound graph is often substantially smaller than the original MC-ECSPP graph. This can lead to quick solution times, but poor upper bounds.

3.1.3 One-node upper bounds: disjoint subsets

We can improve the conservative upper bound by choosing one node within each subset to represent the aggregated node rather than creating an artificial node. Unlike the previous aggregation methods, the one-node upper bound always represents a feasible solution to the original MC-ECSPP.

Two criteria are considered when selecting the representative node: time window length and average travel times. A node with a wide time window and low connecting distances to other subsets is preferred. We evaluate nodes based on their distances to the p closest subsets to eliminate the impact of outliers which will not likely appear in the optimal solution. Let β_i^1 denote the length of the time window of node i ($\beta_i^1 = b_i - a_i; \quad \forall i \in N$) and $\beta_i^2(p)$ denote the average minimum distance from node i to the p closest subsets. We obtain $\beta_i^2(p)$ as follows.

For all $k \in \{1..K\}$ and $i \in N_k$:

(i) $t_{il} = (\min_{j \in N_l} t_{ij} + \min_{j \in N_l} t_{ji})/2 \qquad \forall l \in \{1..K : l \neq k\}.$
if arc (i,j) is not feasible, $t_{ij} = L$

(ii) Sort the t_{il} values in ascending order

(iii) $\beta_i^2(p)$ is the average of the p minimum t_{il} values

Let α be a parameter to weight the relative importance of the two criteria. Subset k is represented by node $j \in N_k$ such that $j = \arg\max_{i \in N_k} \left(\beta_i^1 - \alpha\beta_i^2(p)\right)$. All other nodes are removed from the graph. Based on computational results, we use $p = K/2$ and $\alpha = \frac{\sum_{i \in N_k} \beta_i^1}{\sum_{i \in N_k} (\beta_i^1 + \beta_i^2(p))}$.

Theorem 3 *The optimal solution to the aggregated problem as defined by the one-node upper bound produces a tighter upper bound on the shortest path through the original MC-ECSPP network than the conservative upper bound.*

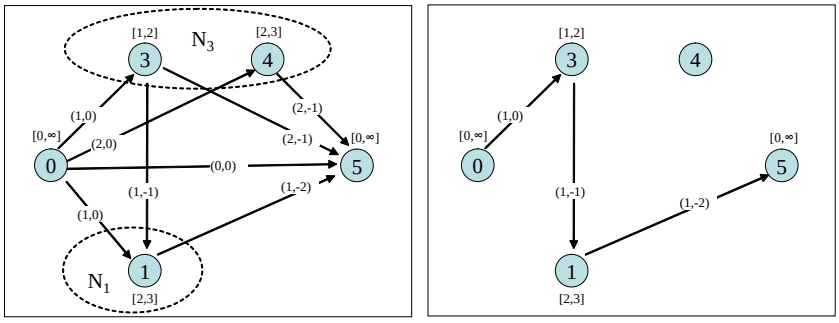
Proof: Let $Z(G)$ denote the cost of the shortest path for the original graph, $G = \{N, A\}$, and let $Z(G^+_{UB_1})$ denote the cost of the shortest path for the one-node aggregated graph, $G^+_{UB_1} =$

$\{N^+_{UB_1}, A^+_{UB_1}\}$. Since $G^+_{UB_1} \subseteq G$, $Z(G^+_{UB_1}) \geq Z(G)$. Let $Z(G^+_{UB_c})$ denote the cost of the shortest path for the conservative upper bound aggregated graph, $G^+_{UB_c} = \{N^+_{UB_c}, A^+_{UB_c}\}$. Let $G' = G^+_{UB_1} \cup G^+_{UB_c}$, whose optimal solution has a cost $Z(G')$. In G', nodes from $G^+_{UB_1}$ dominate nodes from $G^+_{UB_c}$ according to Definition 1 and nodes from $G^+_{UB_c}$ can be eliminated from G', so that $Z(G') = Z(G^+_{UB_1})$. Since $Z(G^+_{UB_c}) \geq Z(G')$, then $Z(G^+_{UB_1}) \leq Z(G_{UB_c})^+$. □

3.2 Expansion subproblem: disjoint subsets

The expansion subproblem transforms the solution path from the aggregated bounding subproblem (lower or upper bound) into a feasible solution to the original MC-ECSPP. With a few modifications noted below, the expansion subproblem is the same for lower and upper bounds. The one-node upper bounds are feasible, yet solutions can be improved by performing an expansion subproblem.

The aggregated graph is expanded to include all nodes within each subset along the shortest path in the aggregated solution. We define the shortest path P^+ for the aggregated problem by the nodes $N(P^+)$ and arcs $A(P^+)$ along the path. The expansion graph $G^- = \{N^-, A^-\}$ is defined as follows. The node set N^- consists of the nodes in the subsets along the solution path: $i \in N_k$ for all $k \in N(P^+)$, and the source and sink nodes. The original time windows are restored for nodes in N^-. The arcs along the shortest path are expanded to connect individual nodes in the subsets with the original travel times and costs. The set A^- consists of arcs $(i, j) \in A$ for all $i \in N_k, j \in N_l$ such that $(k, l) \in A(P^+)$. When expanding a lower bound solution, additional arcs are included to bypass node subsets along the path if no node within the subset is feasible. Since arcs are directed along the direction of P^+ and there are no arcs between nodes within a subset, the expanded graph is acyclic. The choice of nodes visited within the subsets is solved as a CSPP on an acyclic graph.



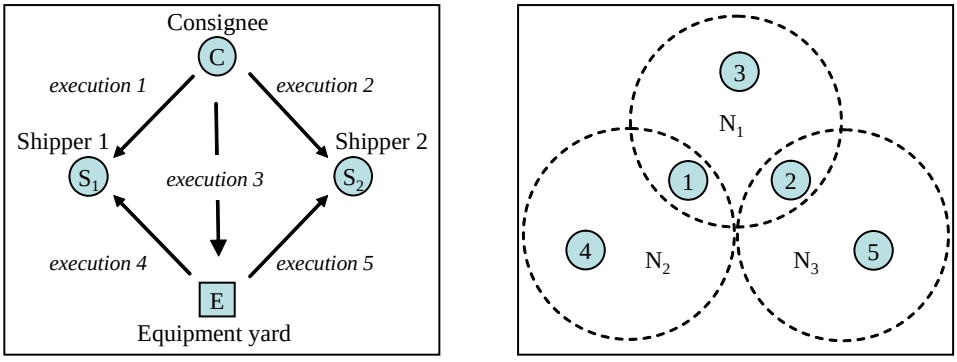
(a) Expanded graph (b) Feasible solution path

Figure 5: Expansion subproblem example: expanding the lower bound from Figure 3

In Figure 5, the aggregated lower bound solution from Figure 3 $(0-[N_3]-[N_1]-5)$ is expanded to obtain a feasible solution to the MC-ECSPP. Nodes from the subset N_2 do not appear since the aggregated node for N_2 is not part of the solution path. The solution to the expansion subproblem is $0-3-1-n$ with a length of 3 units and a cost of -3 units. Since this path represents a feasible solution and the cost is equal to the lower bound, the solution is optimal.

3.3 Non-disjoint node subsets

In this section, the aggregated and expansion subproblems are generalized to allow for non-disjoint node subsets in which a node may appear in one or two subsets. Such non-disjoint subsets often arise when modeling drayage operations. Consider the example in Figure 6(a) with three flexible tasks: (i) Consignee (C) needs to reposition an empty trailer; (ii) Shipper (S_1) needs an empty trailer; and (iii) Shipper (S_2) also needs an empty trailer. The equipment yard which can receive and supply empty trailers. Additionally, the empty trailer from the consignee can be moved directly to one of the shippers, thus eliminating one trailer movement.



(a) Original graph (b) MC-ESCPP graph

Figure 6: Non-disjoint node subsets

In Figure 6(b), the original graph is transformed into an MC-ECSPP graph with non-disjoint subsets. Execution 1 from C to S_1 appears in the subset for the task from the consignee (N_1), since this execution satisfies the need to move an empty trailer, and the subset for shipper 1 (N_2), since this execution also satisfies the need for an empty trailer. Likewise, execution 2 from C to S_2 appears in subsets N_1 and N_3. Executions involving the equipment yard and the consignee and shippers (3,4, and 5) appear in their respective subsets. An execution can satisfy at most two tasks.

Section 3.3.1 presents lower bound subproblems and associated expansion subproblems for non-disjoint node subsets. Section 3.3.2 present upper bound methods.

3.3.1 Lower bounds: non-disjoint subsets

We present two options for incorporating non-disjoint subsets into the lower bound and expansion subproblems by relaxing certain constraints for non-disjoint sets. Recall that the cost of an arc includes the travel time and the benefit of visiting a node: $c_{ij} = t_{ij} + \pi_i$.

Lower bound 1 (LB_1): The first option relaxes the constraint that one execution can satisfy at most two tasks. All intersecting subsets are collapsed into a single node with the benefits (dual values) of all tasks. The same aggregation algorithm for the disjoint case is used; however, instead of one subset, a collection of subsets is collapsed into a single node. Let ϕ_r represent the set of subsets aggregated in a single node r and let N^r represent the set of nodes in r. The time window for r is $[a_r, b_r]$ where $a_r^+ = \min_{i \in N^r} a_i$ and $b_r^+ = \max_{i \in N^r} b_i$. The travel time on arc (r, s) from node r to node s (which may itself be a collection of subsets) is $t_{rs}^+ = min_{i \in N^r, j \in N^s} t_{ij}$. The cost of arc (r, s) is $c_{rs}^+ = min_{i \in N^r, j \in N^s} t_{ij} + \sum_{k \in \phi_r} min_{i \in N_k} \pi_i(k)$, where $\pi_i(k)$ is the portion of the benefit associated with subset N_k from visiting node i. The resulting solution is a lower bound to the original problem since the maximum benefit, $\sum_{k \in \phi_r} min_{i \in N_k} \pi_i(k)$, can be achieved with the minimum resource consumption. The maximum benefit may be infeasible; for example, in Figure 6, if executions 4 and 5 did not exist, it would not be possible to serve all three tasks.

Expanding solution paths from LB_1 can be performed in several ways. In one method, at most one node within each combined subset is chosen. Arcs are created from nodes in a subset to nodes in other subsets along the solution path, but no paths exist to connect nodes within the subset. Therefore, only one node may be selected from each aggregation of subsets even if it is possible to serve additional tasks. This may result in a poor feasible solution if multiple subsets are collapsed into a single subset. In the example in Figure 6, it is possible to serve all three tasks by selecting nodes 4 and 2 or nodes 1 and 5, yet this version of the expansion subproblem would not allow such a solution. Alternatively, more than one node can be chosen as long as the nodes do not satisfy the same original task in the MC-ECSPP. This can be achieved by allowing arcs between nodes within a subset which do not serve the same task. In Figure 6, allowable arcs may be (4,3), (4,2) and (3,4). However, the expansion graph is likely to be cyclic and the problem begins to resemble the original MC-ECSPP in terms of complexity. A compromise between these methods allows arcs between nodes, as long as no cycles are created. The following algorithm creates acyclic graphs. Let $\mathcal{U}$ denote the set of unassigned nodes and $\mathcal{A}$ the set of assigned nodes. To maintain an acyclic graph, arcs are only created between an assigned node and an unassigned node.

For each aggregated node r visited on the solution path:

Step 0: Initialization: $\mathcal{U} = N^r$ and $\mathcal{A} = \emptyset$

Step 1: Order nodes $i \in \mathcal{U}$ by a_i in ascending order (using $b_i - a_i$ as the tie-breaker)

Step 2: Select node i at the top of the list in $\mathcal{U}$:

(i) $\mathcal{A} = \mathcal{A} \cup \{i\}$

(ii) $\mathcal{U} = \mathcal{U} \backslash \{i\}$

(iii) for all $j \in \mathcal{U}$, add arc (i, j) to A^- if feasible

Step 3: Repeat step 2 while $\mathcal{U} \neq \emptyset$

Clearly, the sequence for this algorithm will impact the solution. In Section 3.4, we test LB_1 (a) without arcs between nodes within a subset and (b) with acyclic connections within subsets.

Lower bound 2: (LB_2) The second option relaxes the constraint that connects tasks served by the same execution (an execution from a consignee to a shipper satisfies the task at the shipper and the task at the consignee). Each execution that serves two tasks is duplicated and the copies are placed in disjoint subsets. For the Figure 6 example, execution 1 is copied to form $1'$, and execution 2 is copied to form $2'$. Subset N_1 contains nodes 1, 2, and 3; N_2 contains nodes $1'$ and 4; and N_3 contains nodes $2'$ and 5. The problem is solved as a disjoint lower bound problem.

In the expansion subproblem, the constraints linking tasks are reintroduced in the labeling procedure. An additional label records the tasks served for each candidate path: if a node is visited that serves two tasks, both tasks are recorded in the label when the first node is visited.

3.3.2 Upper bounds: non-disjoint subsets

Non-disjoint subsets can be incorporated easily into the conservative upper bound by removing nodes that appear in multiple subsets. For the one-node upper bound, the node selection problem described in Section 3.1.3 no longer separates by subsets. We introduce the following node selection procedure. Let w_i equal 1 if node i is chosen as a representative node, and 0 otherwise.

$$\max \sum_{i \in N} \left(\beta_i^1 - \alpha \beta_i^2(p)\right) w_i \tag{3a}$$

subject to

$$\sum_{i \in N_k} w_i = 1 \qquad \forall k = 1, , K \tag{3b}$$

$$w_i \in \{0, 1\} \qquad \forall i \in N \tag{3c}$$

The objective function (3a) finds the combination of representative nodes that maximizes the time window and average distance function. Constraints (3b) ensure that a representative node is chosen for each subset. Constraints (3c) define the binary selection variables.

The one-node expansion subproblem combines the expansion methods for LB_1 and LB_2, since issues related to both relaxations must be considered. If a representative node that appears in two subsets is chosen for the aggregated problem, then the expansion graph is constructed such that graph is acyclic, as in LB_1. The expansion subproblem also must include the possibility of choosing a node that appears in two subsets. In this case, the expansion paths must include additional labels to record the tasks served.

3.4 Computational results

The algorithms are implemented using C with the CPLEX Callable Library Interface and the CPLEX 8.1 solver, running on a Sun Fire v250 1.28-GHz UltraSPARC IIIi computer with two processors. Section 3.4.1 describes the test cases. Section 3.4.2 presents the performance of the solution methods for disjoint test cases and Section 3.4.3 presents results for non-disjoint test cases.

3.4.1 Test cases

The algorithms are tested with a set of test cases on a network defined within a 5 by 5 square area. The test cases are grouped by problem characteristics (number of nodes, number of subsets, disjoint or non-disjoint subsets). The number of nodes (executions) and subsets (tasks) for the test cases range from 25 to 150 and 5 to 25, respectively. The characteristics for the test cases are shown in Table 2. Ten randomly generated instances are created for each group of characteristics. Travel times between nodes, t_{ij}, are based on Euclidean distances in the network. The total travel time on a path is limited to 10 units. The values of π_i, the benefit of visiting node $i \in N$, are generated randomly from a uniform distribution between 0 and 10 to calculate $c_{ij} = t_{ij} + \pi_i$. Time windows at the nodes are chosen randomly for a set distribution of time windows.

3.4.2 Performance of MC-ECSPP solution methods: disjoint test cases

The solution methods for the MC-ECSPP are evaluated relative to exact solutions obtained with the method from Smilowitz (2006), which can obtain optimal solutions for the smaller test cases.

Figure 7 plots the average optimality gaps for the bounding and solution methods with disjoint subsets: lower bound, LB, and expansion, $e(LB)$; conservative upper bound, UB_c, and expansion,

Disjoint test cases			Non-disjoint test cases		
Group	Nodes	Subsets	Group	Nodes	Subsets
1	25	5	16	25	5
2	25	10	17	25	10
3	50	5	18	50	5
4	50	10	19	50	10
5	50	15	20	50	15
6	100	5	21	100	5
7	100	10	22	100	10
8	100	15	23	100	15
9	100	20	24	100	20
10	100	25	25	100	25
11	150	5	26	150	5
12	150	10	27	150	10
13	150	15	28	150	15
14	150	20	29	150	20
15	150	25	30	150	25

Table 2: Characteristics for randomly generated test cases

$e(UB_c)$; one-node upper bound, UB_1, and expansion, $e(UB_1)$. Hollow symbols represent aggregated subproblem solutions and solid symbols represent expansion subproblem solutions. Each symbol represents the average over the ten test cases in the group. The vertical lines separate the groups by the number of nodes in the test cases. Table 11 in Appendix A lists these averages and the standard deviation of the ten test cases in each group. Table 3 presents the solution speed and the ability to solve test cases completely for these methods. For each group, the average solution time (over all ten test cases) is shown in CPU seconds. The percent of test cases solved completely is shown as well. The first method, MC, is the exact method for the MC-ECSPP from Smilowitz (2006). As the numbers of nodes and subsets increase, solving to optimality is often not possible. Optimal solutions are found for 90% of the instances from group 8, 40% from group 9, 60% from group 10, 80% from group 12, and 20% from group 13. Optimal solutions cannot be obtained for test cases in groups 14 and 15; no gaps are presented for groups 14 and 15 in Figure 7.

Across all groups, the conservative upper bound and related expansion method are significantly outperformed by the one-node upper bound and related expansion methods. Expanding the solutions from UB_1 with $e(UB_1)$ improves the objective by an average of 12% across all test cases without a significant increase in solution time. All UB_1 and $e(UB_1)$ solutions are obtained in less than one CPU second. While the solution times for LB are less than the solution times of the full

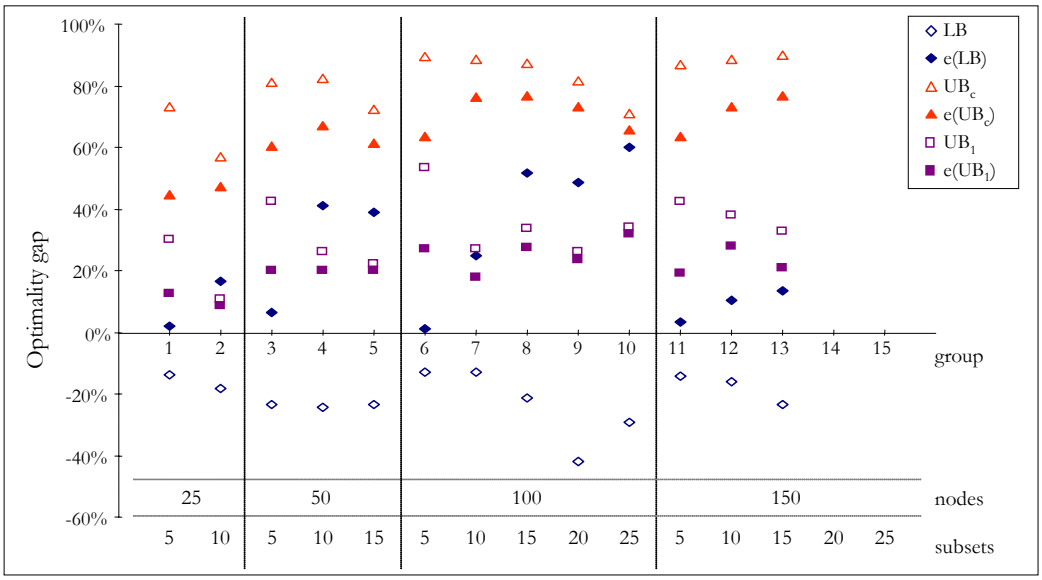


Figure 7: Average optimality gaps for bounds and feasible solutions: disjoint test cases

MC method, several test cases in groups with many nodes and subsets cannot be solved completely due to memory limits. In cases in which the lower bound is not completed, it is still possible to obtain feasible solutions for the best paths obtained (which may not be optimal) using $e(LB)$.

The figure shows a slight trend of increased optimality gaps for UB_1 and $e(UB_1)$ as the number of nodes increase. More noticeable is the improvement in optimality gaps for UB_1 as the number of nodes per subset decreases. Fewer nodes per subset often translates to less diversity in node characteristics (time windows, cost and travel times); therefore, choosing one node to represent the subset is less costly. As the number of nodes per subset increases, the diversity in characteristics increases and it becomes more difficult for one node to adequately represent the subset. The gaps for $e(UB_1)$ are less effected by the number of nodes per subset. For LB and $e(LB)$, optimality gaps increase as the number of nodes per subset decreases.

These trends are studied more in Table 4, which presents a comparison of the solutions obtained with $e(LB)$ and $e(UB_1)$ for test cases grouped by the ratio of nodes to subsets. For each ratio, the table lists the number of nodes per subset and the groups and number of instances represented in that category. For $e(LB)$ and $e(UB_1)$, respectively, the average optimality gap over all instances and the standard deviation are shown. For each ratio, the preferred solution method (either $e(LB)$, $e(UB_1)$, or same) is calculated as the percent of instances in which the respective solution method outperforms the other method. For a ratio of 4, only two instances are solved to optimality and for ratios of 6 and 8, no optimal solutions are found.

	MC		LB		e(LB)		UB_1		$e(UB_1)$		UB_c		$e(UB_c)$	
Group	time	solved	time	solved	time	solved	time	solved	time	solved	time	solved	time	solved
1	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%
2	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%
3	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%
4	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%
5	4	100%	0.1	100%	0	100%	0	100%	0	100%	0	100%	0	100%
6	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%
7	9	100%	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%
8	338	90%	2	100%	0	100%	0	100%	0	100%	0	100%	0	100%
9	1,762	60%	77	60%	0	100%	0	100%	0	100%	0	100%	0	100%
10	4,683	40%	886	10%	0.1	100%	0.3	100%	0	100%	0	100%	0	100%
11	1	100%	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%
12	329	80%	0	100%	0.3	100%	0	100%	0	100%	0	100%	0	100%
13	1,728	20%	9	90%	0	100%	0	100%	0	100%	0	100%	0	100%
14	10,354	0%	633	10%	0	100%	0.1	100%	0	100%	0	100%	0	100%
15	6,664	0%	4,871	0%	0	100%	3	100%	0	100%	0	100%	0	100%

Table 3: Solution times for disjoint problem instances

The results in Table 4 suggest that with fewer nodes per subset (8 or less), $e(UB_1)$ outperforms $e(LB)$, as evidenced by the optimality gaps and the preferred method calculations. With more nodes per subset, $e(LB)$ becomes the preferred method. As the number of nodes per subset increases, $e(LB)$ benefits from additional diversity in characteristics since the aggregated node assumes the best combination of parameters. This results in greater flexibility for the expansion subproblem, which improves the optimality gap of resulting feasible solutions, but also increases the computational effort. For example, the time windows are likely to be wider. As a result, the lower bound solution paths include more nodes with wider time windows.

The above trends are true of randomized cases in which nodes within a subset can be quite heterogenous in terms of time windows, costs and travel distances. In reality, nodes within subsets may have similar attributes (time windows, dual values). In these cases, the number of nodes in a set may be less important.

3.4.3 Performance of solution method for MC-ECSPP: non-disjoint test cases

Figure 8 presents the average optimality gaps with non-disjoint subsets: lower bound 1, LB_1 and expansion options a and b, $e(LB_1, a)$ and $e(LB_1, b)$; lower bound 2, LB_2, and expansion, $e(LB_2)$; one-node upper bound, UB_1 and expansion $e(UB_1)$. Expansion options for LB_1 include no arcs

Nodes/			e(LB)		e(UB_1)		Preferred method		
subset	groups	instances	average	st dev	average	st dev	e(LB)	e(UB_1)	same
3	2,5	20	28%	19%	14%	10%	15%	80%	5%
4	10	10	60%	15%	32%	8%	20%	80%	0%
5	1,4,9	30	28%	26%	18%	16%	17%	67%	16%
6	15	10					10%	90%	0%
7	8	10	52%	17%	28%	5%	10%	90%	0%
8	14	10					30%	70%	0%
10	3,7,13	30	15%	14%	19%	14%	50%	40%	10%
15	12	10	10%	11%	28%	10%	100%	0%	0%
20	6	10	1%	2%	27%	21%	80%	0%	20%
30	11	10	4%	8%	19%	20%	80%	10%	10%

Table 4: Quality of feasible solutions by node per subset ratio: disjoint test cases

within subsets ($e(LB_1, a)$) and acyclic arcs within subsets ($e(LB_1, b)$). The conservative upper bound and its expansion are not shown; as with the disjoint case, the results are significantly worse than those for other methods. Table 12 in Appendix A lists the averages and standard deviations. Table 5 presents the solution times for the non-disjoint test cases. Optimal solutions are found for 90% of the instances from group 23, 60% from group 24, 20% from group 25, 80% from group 27, and 30% from group 28. Optimal solutions can not be obtained for test cases in groups 29 and 30.

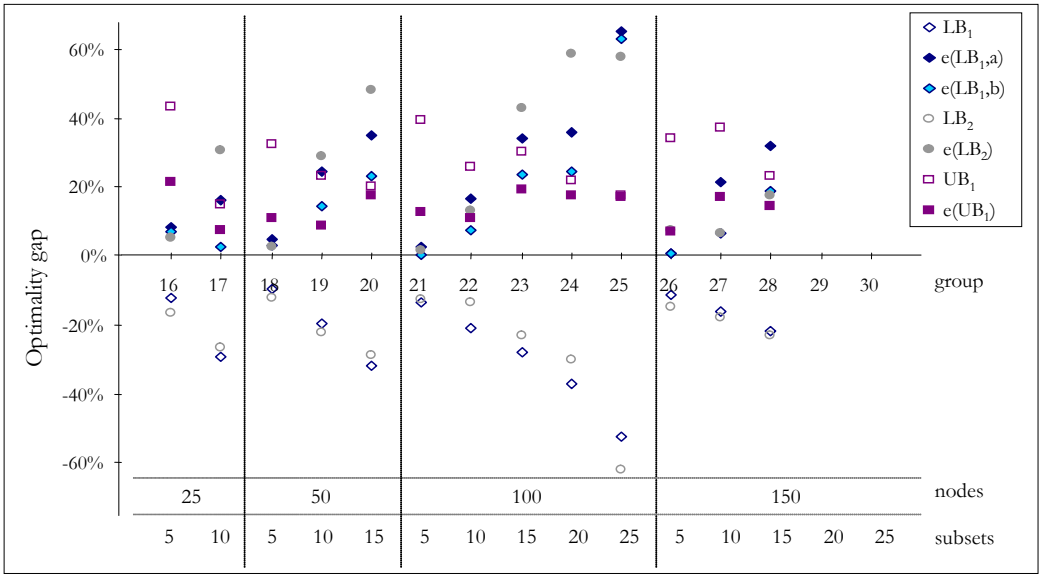


Figure 8: Average optimality gaps for bounds and feasible solutions: non-disjoint test cases

Trends similar to those for disjoint test cases are observed in the optimality gaps and solution times for UB_1 and $e(UB_1)$. The expansion subproblem improves UB_1 solutions by 13% on aver-

	MC		LB_1		$e(LB_1, a)$		$e(LB_1, b)$		LB_2		$e(LB_2)$	
Group	Time	Solved	Time	Solved	Time	Solved	Time	Solved	Time	Solved	Time	Solved
16	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%
17	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%
18	0	100%	0	100%	0	100%	0	100%	0	100%	0	100%
19	0.3	100%	0	100%	0	100%	0	100%	0.1	100%	0	100%
20	1.1	100%	0	100%	0	100%	0	100%	0.9	100%	0	100%
21	0.1	100%	0	100%	0	100%	0	100%	0	100%	0	100%
22	27	100%	0	100%	0	100%	0.1	100%	0	100%	0	100%
23	423	90%	0	100%	0.1	100%	0.3	100%	33	80%	0	100%
24	204	60%	0	100%	0	100%	8.3	100%	266	30%	0.1	100%
25	4805	20%	0.6	100%	0	100%	0	100%	1410	10%	0	100%
26	0.5	100%	0	100%	0	100%	0	100%	0	100%	0	100%
27	552	80%	0	100%	0.1	100%	1	100%	0	100%	0.2	100%
28	2725	30%	0	100%	0.1	100%	282	100%	12	90%	0.1	100%
29	3907	0%	0	100%	0.7	100%	438	100%	91	10%	0	100%
30	9743	0%	0	100%	0.1	100%	183	100%	1664	0%	0	100%

Table 5: Solution time (in seconds) and unsolved instances: non-disjoint test cases

age. The two lower bounds methods yield comparable results, yet the solution times for LB_2 are significantly higher. Further, LB_2 is solved to completion for 80% of the instances from group 23, 30% from group 24, 10% from group 25, 90% from group 28, and 10% from group 29. No solutions are completed for test cases in group 30. All instances are solved with LB_1. The inclusion of acyclic arcs between nodes within subsets in $e(LB_1, b)$ leads to a 14% improvement, on average, over $e(LB_1, a)$. However, the solution times for option b are significantly higher than those for option a.

Nodes/			$e(LB_1, b)$		e(UB_1)		Preferred method		
subset	groups	instances	average	st dev	average	st dev	$e(LB_1, b)$	e(UB_1)	same
3	17,20	20	13%	20%	13%	10%	55%	40%	5%
4	25	10	63%	2%	17%	3%	0%	100%	0%
5	16,19,24	30	14%	17%	16%	13%	47%	43%	10%
6	30	10					50%	50%	0%
7	23	10	24%	12%	19%	11%	50%	50%	0%
8	29	10					70%	30%	0%
10	18,23,28	30	7%	8%	12%	12%	53%	40%	7%
15	27	10	6%	10%	17%	12%	80%	20%	0%
20	21	10	0%	0%	13%	12%	70%	0%	30%
30	26	10	1%	2%	7%	10%	50%	0%	50%

Table 6: Quality of feasible solutions by node per subset ratio: non-disjoint test cases

We compare $e(LB_1, b)$ and $e(UB_1)$ in Table 6 as a function of the number of nodes per subset. The trends from the disjoint test cases are less strong in the non-disjoint test cases. With less than 8 nodes/subset, $e(UB_1)$ slightly outperforms $e(LB_1, b)$; with more than 8 nodes/subset, $e(LB_1, b)$ outperforms $e(UB_1)$. As in the disjoint cases, the $e(UB_1)$ solutions are less impacted by the number of nodes per subset.

3.4.4 Observations

Based on the computational study, we make the following observations about the solution methods.

- For both disjoint and non-disjoint test cases, the one-node upper bound and its expansion are good options for quick solutions. While UB_1 itself is a feasible solution, using $e(UB_1)$ to expand solutions from UB_1 leads to improved solutions while not increasing solution times significantly.
- For both disjoint and non-disjoint test cases, the lower bounds perform well with many nodes per subset. The optimality gaps increase for larger instances with fewer nodes per subset.
- For non-disjoint test cases, lower bound 1 is significantly faster than lower bound 2 and can solve larger instances. Optimality gaps for these methods are not significantly different. Increasing the arc set in the expansion subproblem for LB_2 improves the optimality gap, but increases the solution time.

4 Implementing the MC-ECSPP in branch and price for the MRRP

We incorporate the bounding and solution methods for the MC-ECSPP from the previous section in a branch-and-price solution method for the MRRP. Section 4.1 describes the solution approach and Section 4.2 presents the computational results and analysis.

4.1 Solution method

We use branch and price to obtain integer solutions for the MRRP, originally proposed in Smilowitz (2006). The linear relaxation of (1) is solved at each node of the branch-and-price tree. If the solution to (1) is integer at a node, the upper bound may be updated. Otherwise, if the solution is greater than or equal to the upper bound, the node is truncated; if not, we branch on the route variable closest to 1.

4.1.1 Pricing problem: exact solution method

At each node, the linear relation of (1) is solved with column generation to add routes to the restricted route set $\mathcal{R}'$. The current dual values associated with constraints (1b) are used in the pricing problem (which is an MC-ECSPP) to generate routes for $\mathcal{R}'$. In what follows, we describe how the methods from Section 3 are used to solve the MC-ECSPP in stages, beginning with the fastest solution method, and calling more intensive methods as needed, as shown in Figure 9.

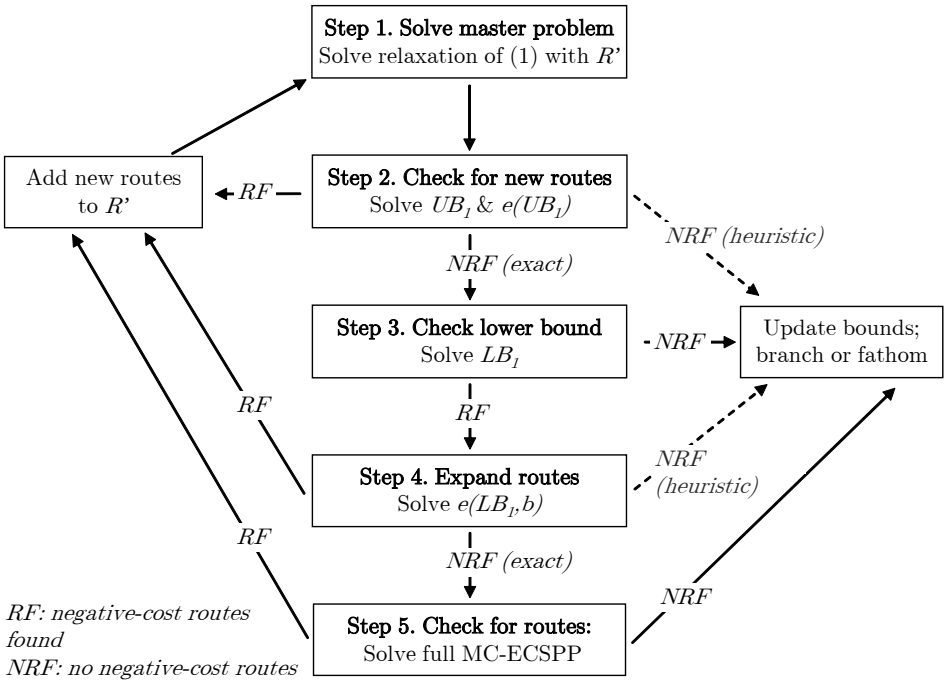


Figure 9: Solution method at a node of the branch-and-price tree for MRRP

In Step 1, the initial route set, $\mathcal{R}'$, is found at the root node as follows. Each task is assigned to its own route. For flexible tasks, a default execution is chosen by picking the closest equipment yard. In Step 2, the first check for new routes is performed with UB_1 and $e(UB_1)$. The UB_1 method may produce several negative reduced cost paths as a result of the label correcting solution method. Since $e(UB_1)$ is quick, we expand many of these routes. If routes with negative reduced costs are found with $e(UB_1)$, they are added to $\mathcal{R}'$ and the linear relaxation of (1) is resolved in Step 1. Otherwise, the search for routes continues in Step 3 with LB_1. If the optimal solution to LB_1 is not negative, then the current solution to the linear relaxation of (1) is optimal since the MC-ECSPP contains no negative reduced cost routes. If the optimal solution to LB_1 is negative, this route is expanded in Step 4 with $e(LB_1, b)$. If the expanded path has a negative reduced cost,

it is added to $\mathcal{R}'$ and the linear relaxation of (1) is resolved in Step 1. If the expanded path does not have a negative reduced cost, we cannot conclude that the current solution to the linear relaxation of (1) is optimal since the expansion subproblem is a heuristic. The full MC-ECSPP is then solved without node aggregation in Step 5 to determine if negative cost routes exist. This method may produce several negative reduced cost paths which may be added to $\mathcal{R}'$.

As noted, the MC-ECSPP solution methods may produce multiple routes with negative reduced costs. Rather than adding only the route with the lowest cost, experimental results suggest that adding a maximum of 100 routes effectively balances the number of column generation iterations required with the total number of columns added.

4.1.2 Pricing problem: heuristic variations

We propose two heuristic variations of the exact approach. The variations involve removing or limiting the most time-consuming steps of the exact solution approach, shown in dashed lines in Figure 9. The first option skips Step 5 in which the full MC-ECSPP is solved; in Step 4 if no negative reduced cost routes are obtained, we continue to traverse the branch-and-price tree. The second option removes Steps 3, 4 and 5, and generate routes only with UB_1 and $e(UB_1)$.

We also apply stopping criteria: maximum iteration limit, acceptable solution gap, and limited solution improvement. First, the column generation procedure is terminated at a branch-and-price node if the number of iterations exceeds 25 or if the objective value does not improve by more than 10^{-5} after 10 iterations. Further, the number of route possibilities for instances with many flexible tasks may lead to an exorbitant number of nodes to explore. We iteratively relax the truncation criteria within the branch-and-price approach as the solution time and the number of nodes explored increase. Typically, a node i is truncated if z_i, the solution to linear relaxation of (1), is greater than or equal to z_{UB}, the upper bound. As in Smilowitz (2006), we introduce ϵ_i such that a node is truncated if $z_i \geq (1 - \epsilon_i) z_{UB}$. At the root node, we set $\epsilon_0 = 0$. At each subsequent node i, ϵ_i slowly increases with the number of nodes and solution time. As a result, the branch-and-price algorithm runs to completion for small instances, but not for larger instances.

4.2 Computational results

Section 4.2.1 describes the test cases. Sections 4.2.2, 4.2.3, and 4.2.4 analyzes the performance of the exact method and heuristic variations 1 and 2, respectively.

4.2.1 Test cases

The test cases are based on data from drayage and third party logistics companies (Dahnke (2003); Corinescu (2003); Grosz (2003)) for dray operations over a region including greater Chicagoland and parts of central Illinois, southern Wisconsin and western Indiana. We use an aggregated data set based on the properties of the original proprietary customer data from several drayage companies. The distance matrices for the aggregated data sets maintain the same geographical characteristics as the initial industry data. Table 7(a) presents the number of total tasks, flexible tasks and fixed tasks for the test cases.

Test case	Tasks	Flexible	Fixed
1	25	13	12
2	25	10	15
3	25	10	15
4	25	10	15
5	25	19	6
6	50	27	23
7	50	23	27
8	50	31	19
9	50	24	26
10	50	25	25
11	75	43	32
12	75	43	32
13	75	42	33
14	75	37	38
15	75	34	41

Parameter	Value
Loaded trailer pick-up time	30 minutes
Loaded trailer drop-off time	30 minutes
Empty trailer pick-up time	15 minutes
Empty trailer drop-off time	15 minutes
Trailer loading time	1 hour
Trailer unloading time	1 hour
Driver work shift	10 hours (continuous)

Table 7: (a) Test cases of computational study of MRRP; (b) Operating parameters

The operating parameters are detailed in Table 7(b). The model captures one day of operation, assuming the loads are known when decisions are made. It is assumed that all tractor routes begin and end at one central depot, and that drivers work a continuous ten-hour work shift.

4.2.2 Exact method results

The exact solution method can solve only test cases 1, 3, 4, and 5 to optimality. Table 8 shows the numerical results for these test cases. We present the two objective functions (*fleet size* and *travel distance*), along with the solution time in CPU seconds. The table also shows the total number of branch-and-price nodes generated in the tree (*B&P nodes*), and the average number of column generation iterations performed at each node (*Iterations per node*). Additionally, we present the

average number of times that Step 5 is performed at each node (*Step 5 per node*). The final four columns present the total number of columns generated and the percentage of columns generated at each step of the column generation approach.

			Solution	B&P	Iterations	Step 5	Columns generated			
Test case	Fleet	Distance	time	nodes	per node	per node	Total	Step 2	Steps 3,4	Step 5
1	12	114	25	103	4.0	1.0	780	77%	23%	1%
3	17	148	6	111	3.7	1.0	533	79%	21%	0%
4	16	153	13	107	4.2	1.1	740	58%	27%	15%
5	18	163	2	13	7.1	1.7	582	72%	25%	3%

Table 8: Exact solution method results for small test cases

While the solution times in Table 8 are small, the times grow prohibitively large for the other test cases, particularly the time required for Step 5. As the table indicates, Step 5 is called between 1-2 times per node, making these test cases easier to solve. The contribution to the total number of columns from Step 5 is minimal in all cases, except test case 4. The majority of columns is generated in Step 2 (71% on average) and Steps 3 and 4 (24% on average), suggesting that removing Step 5 in the heuristic will not have a large impact on the objective function.

4.2.3 Heuristic variation 1 results

The first heuristic variation removes Step 5 from the solution approach. Additionally, when the solution time for LB_1 exceeds 1,000 CPU seconds, we impose a limit on the size of the problem, restricting the number of subsets to 50 (those with the most negative dual values). This limit is relaxed in subsequent iterations if the solution time is less than 1,000 CPU seconds. Note that this restriction is needed only for test case 6.

Table 9 shows the results for the first heuristic variation. The results are compared with the optimal solutions from Table 8, when possible. As the table shows, four additional test cases can be solved with the heuristic.

The heuristic produces optimal or near-optimal solutions for the first four test cases. Note that the heuristic yields the optimal solution for test case 4, which has the highest percentage of columns generated in Step 5 in Table 8. The solution times and branch-and-price nodes are similar to those with the exact method. Solution times increase significantly for the larger test cases. Steps 3 and 4 contribute a large percentage of the total columns in most cases; however, solving LB_1 in Step 3 is far more time consuming than solving UB_1 in Step 2.

	Exact solution		Heuristic 1		Solution	B&P	Iterations	Column generation		
Test case	Fleet	Distance	Fleet	Distance	time	nodes	per node	Total	Step 2	Steps 3,4
1	12	114	12	114	28	107	3.6	1,351	62%	38%
3	17	148	17	148	8	113	2.6	561	76%	24%
4	16	153	16	153	18	107	3.4	829	60%	40%
5	18	163	18	164	2	3	8	574	82%	18%
6	-	-	23	226	49,002	115	4.2	2,333	97%	3%
7	-	-	27	261	502	117	4.6	2,900	65%	35%
9	-	-	24	237	642	115	3.8	2,220	73%	27%
10	-	-	25	243	475	11	7.4	1,293	70%	30%

Table 9: Heuristic variation 1 results

4.2.4 Heuristic variation 2 results

The second heuristic variation omits Step 3, 4 and Step 5, and only UB_1 and $e(UB_1)$ are applied to generate routes. The stopping criteria described in Section 4.1.2 are also applied. Table 10 presents the results for the second heuristic, compared with the results from the first heuristic. All test cases can be solved with the second heuristic.

Test	Heuristic 1		Heuristic 2		Solution	B&P	Iterations	
case	Fleet	Distance	Fleet	Distance	time	nodes	per node	Total columns
1	12	114	13	122	19	255	1.2	1,248
2	-	-	13	118	106	2,103	1.1	1,379
3	17	148	17	152	7	159	1.3	486
4	16	153	17	160	9	105	1.6	766
5	18	164	18	171	1	3	4	471
6	23	226	26	241	116	7	3.1	793
7	27	261	28	265	101	117	1.8	1,573
8	-	-	28	281	1,211	109	1.9	1,959
9	24	237	25	240	120	115	1.9	1,970
10	25	243	27	257	10	1	12	777
11	-	-	39	383	2,381	129	1.9	2,333
12	-	-	44	433	1,135	123	1.9	2,900
13	-	-	48	449	5,714	119	1.9	8,302
14	-	-	38	379	590	123	1.9	2,220
15	-	-	40	379	7,603	125	1.8	2,221

Table 10: Heuristic variation 2 results

On average, the second heuristic yields solutions that are less than 5% higher than the solutions with heuristic 1, in terms of both fleet size and travel distance. The solution times are significantly lower for those test cases solved with both heuristic variations, resulting in an average decrease of

64%. Note that instances requiring fewer branch-and-price nodes have more iterations per node, since more iterations are typically needed at higher levels of the tree.

5 Conclusions

In this paper, we present a new variation of the elementary constrained shortest path problem, referred to as the Multiple Choice Elementary Constrained Shortest Path Problem (MC-ECSPP). We develop bounding and solution methods for the MC-ECSPP and incorporate these methods into a branch-and-price algorithm for the Multi-Resource Routing Problems. As this is the first step in the study of the MC-ECSPP, we envision several directions of future work. In particular, future work may consider additional solution methods for the MC-ECSPP, with a focus on those methods which are better suited for branch-and-price methods.

Acknowledgment

This research has been supported by grant DMI-0348622 from the National Science Foundation and the Alfred P. Sloan Foundation.

References

Corinescu, E. (2003). Drayage data files from 2002. Hub City Terminals, Inc.

Crainic, T. and Florian, M. (2005). Computing shortest paths with logistic constraints. Working paper.

Dahnke, B. (2003). Drayage data files from 1998. Laser Trucking, personal communication.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, **40**(2), 342–354.

Desrosiers, J., Dumas, Y., Solomon, M., and Soumis, F. (1995). Time constrained routing and scheduling. In *Handbooks in operations research and management science*, volume 8, pages 35–140. Elseiver science B.V.

Dror, M. (1994). Note on the complexity of the shortest path problem with resource constraints for column generation in VRPTW. *Operations Research*, **42**, 977–978.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, **54**, 7–22.

Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2004). An exact algorithm for elementary shortest path problems with resource constraints: application to some vehicle routing problems. *Networks*, **44**(3), 216–229.

Grosz, J. (2003). Personal communications. Cushing Transportation , Inc.

Irnich, S. and Desaulniers, G. (2004). Shortest path problems with resource constraints. Technical Report Les Cahiers du GERAD G-2004-11, École des Hautes Études Commerciales, Montréal.

Irnich, S. and Villeneuve, D. (2004). The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$.

Kolen, A., Rinnooy-Kan, A., and Trienekens, H. (1987). Vehicle routing with time windows. *Operations Research*, **35**(2), 266–273.

Savelsbergh, M. and Sol, M. (1998). DRIVE: Dynamic routing of independent vehicles. *Operations Research*, **46**, 474–490.

Smilowitz, K. (2006). Multi-resource routing with flexible tasks: an application in drayage operations. *IIE Transactions*, **38**(7), 555–568.

Villeneuve, D. and Desaulniers, G. (2005). The shortest path problem with forbidden paths. *European Journal of Operational Research*, **165**, 97–107.

A MC-ECSPP tests

Group	LB	e(LB)	UB_c	$e(UB_c)$	UB_1	$e(UB_1)$
1	-14% (22%)	2% (3%)	73% (16%)	44% (27%)	30% (19%)	12% (20%)
2	-18% (11%)	17% (11%)	57% (12%)	47% (15%)	11% (5%)	8% (7%)
3	-24% (18%)	6% (7%)	81% (11%)	60% (17%)	42% (19%)	20% (17%)
4	-24% (7%)	41% (22%)	82% (5%)	67% (11%)	26% (13%)	20% (13%)
5	-24% (9%)	39% (20%)	72% (9%)	61% (8%)	22% (11%)	20% (10%)
6	-13% (11%)	1% (2%)	89% (5%)	63% (22%)	53% (15%)	27% (21%)
7	-13% (4%)	25% (14%)	88% (6%)	76% (8%)	27% (9%)	18% (8%)
8	-21% (9%)	52% (17%)	87% (4%)	77% (8%)	34% (4%)	28% (5%)
9	-42% (13%)	49% (15%)	82% (5%)	73% (4%)	26% (9%)	23% (8%)
10	-29% (0%)	60% (15%)	71% (8%)	66% (7%)	34% (7%)	32% (8%)
11	-14% (12%)	4% (8%)	87% (7%)	64% (16%)	42% (20%)	19% (20%)
12	-16% (8%)	10% (11%)	89% (3%)	73% (13%)	38% (11%)	28% (10%)
13	-24% (10%)	13% (11%)	90% (0%)	77% (4%)	33% (16%)	21% (17%)

Table 11: Average optimality gaps for bounds and feasible solutions: disjoint test cases. Standard deviations shown in parentheses.

Group	LB_1	$e(LB_1a)$	$e(LB_1, b)$	LB_2	$e(LB_2)$	UB_c	$e(UB_c)$	UB_1	$e(UB_1)$
16	-12% (16%)	8% (13%)	7% (12%)	-17% (25%)	5% (11%)	75% (15%)	47% (27%)	43% (18%)	22% (17%)
17	-29% (15%)	16% (13%)	3% (4%)	-27% (13%)	31% (24%)	67% (16%)	45% (19%)	15% (9%)	8% (7%)
18	-10% (8%)	5% (7%)	3% (6%)	-12% (7%)	2% (4%)	83% (9%)	54% (20%)	32% (28%)	11% (16%)
19	-20% (10%)	24% (19%)	14% (18%)	-22% (12%)	29% (24%)	80% (7%)	66% (11%)	23% (14%)	9% (10%)
20	-32% (11%)	35% (23%)	23% (25%)	-29% (15%)	48% (20%)	70% (8%)	54% (7%)	20% (10%)	18% (10%)
21	-13% (15%)	2% (5%)	0% (0%)	-13% (14%)	1% (2%)	86% (9%)	59% (12%)	39% (16%)	13% (12%)
22	-21% (19%)	17% (11%)	7% (7%)	-13% (6%)	13% (11%)	89% (5%)	65% (19%)	26% (11%)	11% (9%)
23	-28% (11%)	34% (8%)	24% (12%)	-23% (11%)	43% (17%)	85% (3%)	67% (12%)	30% (8%)	19% (11%)
24	-37% (14%)	36% (15%)	24% (14%)	-30% (6%)	58% (19%)	86% (5%)	73% (6%)	22% (4%)	18% (4%)
25	-52% (12%)	65% (1%)	63% (2%)	-62% (0%)	58% (7%)	79% (3%)	67% (8%)	17% (3%)	17% (3%)
26	-12% (17%)	1% (2%)	1% (2%)	-15% (14%)	8% (14%)	87% (6%)	58% (15%)	34% (8%)	7% (10%)
27	-16% (10%)	21% (12%)	6% (10%)	-18% (10%)	7% (9%)	91% (2%)	72% (10%)	37% (11%)	17% (12%)
28	-22% (13%)	32% (4%)	19% (5%)	-23% (14%)	17% (5%)	91% (2%)	77% (6%)	23% (9%)	14% (5%)

Table 12: Average optimality gaps for bounds and feasible solutions: non-disjoint test cases. Standard deviations shown in parentheses.