Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)

A new meta-heuristic for the Bus Driver Scheduling Problem: GRASP combined with Rollout

Gionatan D'Annibale* Renato De Leone† Paola Festa‡ Emilia Marchitto§

Abstract

The Bus Driver Scheduling Problem (BDSP) is one of the most important planning decision problems that public transportation companies must solve and that appear as an extremely complex part of the general Transportation Planning System. It is formulated as a minimization problem whose objective is to determine the minimum number of driver shifts, subject to a variety of rules and regulations that must be enforced, such as overspread and working time.

In this article, a greedy randomized adaptive search procedure (GRASP) and a Rollout heuristic for BDSP are proposed and tested. A new hybrid heuristic that combine GRASP and Rollout is also proposed and tested. Computational results indicate that these randomized heuristics find near-optimal solutions.

I Introduction

Rollout is a new meta-heuristic recently proposed by (1) for hard combinatorial optimization problems. In this paper, to efficiently determine good quality solutions for the Bus Driver Scheduling Problem (BDSP), we study and test the performance of a Rollout-type algorithm and compared it with a GRASP (Greedy Randomized Adaptive Search Procedure). Moreover, we propose an innovative hybrid meta-heuristic that combines GRASP with Rollout for the problem under examination.

The goal of this paper is to determine optimal shifts (i.e., a full day of work) for local public transportation companies. The BDSP is formulated as a minimization problem whose objective is to determine the minimum number of driver shifts necessary to cover a set of Pieces-Of-Work, subject to a variety of rules and regulations that must be enforced, such as overspread and working time. This problem is one of the most important planning decision problems that public transportation companies must solve and that appear as an extremely complex part of the general Transportation Planning System ((2; 3)).

Since the terms used in literature in dealing with the BDSP are not uniform, it is functional to define the terminology adopted in this paper. For the Vehicle Scheduling Problem, trips are defined as a one-way travel of a bus between two points (i.e., locations in the same city or in different cities), and a running board is a sequence of trips assigned to the same bus. The terms Piece-Of-Work and shift are introduced into the Crew Scheduling Problem context (in our case Bus Driver Scheduling Problem). A Piece-Of-Work is a part of a shift that covers a set of consecutive trips in a running board, subject to different rules and regulations. A set of Pieces-Of-Work that satisfies all the constraints is a feasible shift. Then, the Bus Driver Scheduling Problem consists in assigning the Pieces-Of-Work to shifts such that each Piece-Of-Work is performed by only one driver, the shifts are feasible (that depends on a specified set of working rules), and the total operational costs of the shifts and the total number of shifts are minimized.

The combinatorial nature of the Crew Scheduling Problem and the large dimension of real-world instances have led to the development of sev-

*G. D'annibale is Master student at the Mathematics and Computer Science Dept, University of Camerino, Camerino (MC), Italy (e-mail: gionatan.dannibale@studenti.unicam.it).

†R. De Leone is Full Professor at the Mathematics and Computer Science Dept, University of Camerino, Camerino (MC), Italy (e-mail: renato.deleone@unicam.it).

‡P. Festa is Assistant Professor at the Mathematics and Applications Dept, University of Napoli FEDERICO II, Compl. M.S.A., Via Cintia, 80126 Napoli, Italy (Corresponding Author - phone: +39-081-675605; fax: +39-081-675605; e-mail: paola.festa@unina.it).

§E. Marchitto is PhD at the Mathematics and Computer Science Dept, University of Camerino, Camerino (MC), Italy (e-mail: emilia.marchitto@unicam.it).

eral exact and heuristic algorithms. (4) provided an outline of the Bus Driver Scheduling Problem (BDSP) and proposed various approaches for its solution. Many of these techniques have been reported in proceedings of international conferences on Computer-Aided-Scheduling of Public Transport (e.g., (5), (6), (7), (8), (9), (10) and (11)).

The remaining of the paper is organized as follows. In Section II, the construction and local search phases of a GRASP for the Bus Driver Scheduling Problem are described. Section III presents the Rollout procedure, which is then applied to the problem under examination. Section IV shows how GRASP and Rollout are combined and reports the computational results. Finally, in Section V our concluding remarks and suggestions for future work can be found.

II GRASP for the BDSP

In this section, we provide a brief description of the construction and local search phases of GRASP for the Bus Driver Scheduling Problem (for more details, see (12) and (13)). Given a set W of Pieces-Of-Work, in the sequel we denote with T a feasible schedule corresponding to a set of feasible shifts and with $c(T)$ the sum of the operational costs of the shifts in schedule T.

A Construction Phase

In the GRASP construction phase a feasible schedule T is built, i.e., a set of feasible shifts T_i, $i = 1, \ldots, N$, adding one Piece-Of-Work at a time in a greedy randomized way, until all Pieces-Of-Work have been assigned. Starting with an empty solution T, iteratively a *Restricted Candidate List* (RCL) is constructed for each Piece-Of-Work according to pairwise compatibility and starting time. A Piece-Of-Work is randomly chosen and then added to the current partial solution. Once a Piece-Of-Work is selected, it must be removed from the current set of candidates and the set of Pieces-Of-Work must be adaptively adjusted to take into account that the newly selected Piece-Of-Work is now part of the current partial solution. Once the current shift is completed, a new shift is constructed. The updating procedure for the candidate list is the computational bottleneck of the construction phase. The procedure ends when all the Pieces-Of-Work have been assigned.

B Local Search

Starting from the feasible solution obtained at the end of the construction phase, a local search procedure is applied in order to guarantee local optimality. Given a feasible solution $T = \{T_1, T_2, \ldots, T_N\}$, for $i \in \{1, 2, \ldots, N\}$, we decompose each T_i in partial consecutive shifts, each having h_i and k_i Pieces-Of-Work, respectively. More in detail, assuming that $h_i \leq k_i$, T_i is decomposed as $T_{i1} = \bigcup_{r=1}^{h_i} POW_r$ and $T_{i2} = \bigcup_{r=h_i+1}^{k_i} POW_r$. A similar decomposition can be applied to T_j to obtain T_{j1} and T_{j2}. The variable neighborhood swap $N(T)$ used in the local search is defined as the following set:

$\{\overline{T} = \{\overline{T}_1, , \ldots, \overline{T}_N\}$ s.t.
$\forall\ i = 1, \ldots, N-1, \forall\ i < j \leq N,\ l = 1, \ldots, N,$
$\forall\ i1 = 1, \ldots, h_i,\ \forall\ i2 = h_i + 1, \ldots, k_i,$
$\forall\ j1 = 1, \ldots, h_j,\ \forall\ j2 = h_j + 1, \ldots, k_j,$
$\overline{T}_l = T_l\ \forall l \neq i, j, \overline{T}_j = T_{j1} \cup T_{i2}, \overline{T}_i = T_{i1} \cup T_{j2}\}$.

A feasible schedule T is a local minimum if and only if $c(T) \leq c(\overline{T})$ for all $\overline{T} \in \mathcal{N}_1(T)$.

III Rollout

Rollout [1] algorithms have been recently proposed by (15), (16), (1), and (17). These new methods for $\mathcal{NP}$-hard combinatorial optimization problems have the advantage of amplifying the efficaciousness of any given heuristic algorithm. However, their computational complexity represents a severe limitation when solving large scale problems. The Rollout algorithm determines a solution of the problem (either minimization or maximization) starting from a partial solution and applying at each step a suboptimal algorithm able to produce feasible solutions (*base heuristic* $\mathcal{H}$). The following representation of a combinatorial optimization problem as a graph search problem, as in (15), will be helpful in describing sequential Rollout algorithms.

[1] The name "Rollout" was introduced by Tesauro ((14)) as a synonym for repeatedly playing out a given backgammon position to calculate by Monte Carlo averaging the expected game score starting from that position.

Let $G = (N, A)$ be a direct graph, where N represents the set of nodes and $A \subseteq N \times N$ the set of arcs; let s be an *origin* node and let $\overline{N} \subseteq N$ be the *destination* nodes and assume that for each element $i \in \overline{N}$ a cost function $g(i)$ is assigned. The aim is to determine a directed path which starts at s (origin) and ends at one of the nodes $i \in \overline{N}$ (destination) such that the cost $g(i)$ is minimized. We assume that N, A and $\overline{N}$, that is the set of nodes, the set of arcs and the set of destination nodes, have a finite number of elements, as normally is the case for many $\mathcal{NP}$-hard combinatorial optimization problems. We assume that there are no parallel arcs in the graph. A directed path is a set of arcs $(i_1, i_2), (i_2, i_3), \ldots, (i_{n-1}, i_n)$ that we indicate as the sequence of nodes $(i_1, i_2, \ldots, i_n)$ for convenience. For each non-destination node $i \in N \setminus \overline{N}$, we assume there is at least one path starting at that node and ending at some destination node. Let $\mathcal{H}$ be a path construction algorithm which, given a non-destination node $i \in (N \setminus \overline{N})$, constructs a direct path $(i, i_1, \ldots, i_m, \bar{k})$ starting at i and ending at one of the destination nodes $\bar{k} \in \overline{N}$. For a non-destination node $i \in N \setminus \overline{N}$, we define the associated cost as $c(i) = g(\bar{k})$, where $\bar{k}$ is the end node of the path, built by the path construction algorithm $\mathcal{H}$; moreover we define $c(\bar{k}) = g(\bar{k})$, for each $\bar{k} \in \overline{N}$. To obtain a suboptimal solution of the problem it is possible to start at s (the origin) and apply the path construction algorithm $\mathcal{H}$ to determine a solution of cost $c(s)$. Another possibility is to build a path to a destination by applying $\mathcal{H}$ in a sequential way. The typical sequence starts from a node i and considers all its downstream neighbors j, to which $\mathcal{H}$ is applied. The neighbor that provides the best result becomes the new starting node. This sequential method to apply $\mathcal{H}$ is called the *Rollout algorithm based on* $\mathcal{H}$ and is referred to as $\mathcal{RH}$. For a formal description of the Rollout algorithm, let $\mathcal{N}(i)$ be the neighborhood of node i defined as $\mathcal{N}(i) = \{j | (i, j) \in A\}$. Since there is at least one path starting at node i and ending at a destination node $\bar{k}$, the neighborhood $\mathcal{N}(i)$ is non empty for each $\bar{k} \in N \setminus \overline{N}$. The Rollout algorithm begins with a degenerate path P made only of the origin node s, i.e., $P = \{s\}$. Every iteration adds a new node to the path and this procedure is repeated until the path reaches a destination node. Let $P_t = (s, i_1, \ldots, i_t)$ be a sequence of nodes where i_t is a non-destination node; the typical iteration t of $\mathcal{RH}$ can be illustrated as follows.

1. Determine the neighborhood $\mathcal{N}(i_t)$ of the final node i_t of the path P_t, that is:
$$\mathcal{N}(i_t) = \{j | (i_t, j) \text{ is an arc}\}.$$
2. For every node j which belongs to $N(i_t)$ calculate the cost $c(j)$ by performing the base heuristic $\mathcal{H}$ on the path resulting from the increase of P_t with j.
3. Select the node i_{t+1} that minimizes the cost in $\mathcal{N}(i_t)$, that is:
$$i_{t+1} = \arg \min_{j \in \mathcal{N}(i_t)} c(j).$$
4. Add the node i_{t+1} to the current path P_t, and set
$$P_{t+1} = P_t \cup \{i_{t+1}\}.$$
The Rollout algorithm stops if the node i_{t+1} is a destination; in this case, the path P_{t+1} is the solution with cost $g(i_{t+1})$. When this not the case, swap P_t with P_{t+1} and continue with another iteration.

From the description of the Rollout algorithm it is clear that the computational cost of all iterations is determined by the number of applications of the base heuristic $\mathcal{H}$ (Step 2), which is related to the dimension of $\mathcal{N}(i_t)$. There are different possible strategies to improve the computational complexity of $\mathcal{RH}$; for more details, we refer interested the reader to (18) and (19).

A A Rollout algorithm for the BDSP

In this subsection, we describe a Rollout algorithm for the Bus Driver Scheduling Problem. We have used the GRASP construction procedure as base heuristic $\mathcal{H}$. Starting from a partial solution, the algorithm constructs a complete solution, for each Piece-Of-Work that can be chosen, maintaining the GRASP's characteristics of greediness, randomness, and adaptiveness. For this reason, the version of the Rollout algorithm we propose is non-deterministic. In particular, $\mathcal{H}$ starts from a partial solution and, at each iteration, the solution is augmented adding a Piece-Of-Work j. Clearly, j must be compatible with the previous Piece-Of-Work belonging to the partial solution and minimizes the cost in terms of objective function value of the augmentation. More in details, after $l-1$ iterations, we have a solution T composed of $l-1$ Pieces-Of-Work and, at the following iteration, we add a new Piece-Of-Work to the shift. The pseudocode of Rollout is depicted in Figure 1. Given the input set of Pieces-of-Work W and a partial schedule T, `Construct`$(L(T), L_{max})$ procedure in line 5 is used to built the list of candidates that must be added to schedule T. The parameter L_{max} is

```
algorithm rollout(W, RandomSeed, L_max, c)
1    T := ∅;
2    k := 1;
3    while (∃POW ∈ W s.t. POW ∉ T)
4        Construct(L(T), L_max);
5        if (L(T) ≠ ∅)
6            B := ∅; c(B) := +∞;
7            for each POW_c ∈ L(T)
8                SC_k := T ∪ POW_c;
9                P := H(SC_k, RandomSeed);
10               if (c(P) < c(B))
11                   B := SC_k;
12               end if
13           end for each
14           T := B;
15       else
16           k := k + 1;
17       end if
18   end while
19   return(T);
end rollout;
```

Figure 1: Rollout procedure for the BDSP.

the cardinality of the list $L(T)$ and is fixed *a priori*. In lines 8 through 14, the procedure starts from a partial solution and applies at each step the base heuristic $\mathcal{H}(SC_k, RandomSeed)$, that corresponds to the GRASP construction phase already described in Section A. The best choice at each Rollout iteration is saved in the variable B, while in POW_c is saved the current element of the list $L(T)$, i.e. a candidate Piece-Of-Work to be added to the partial schedule T. The value of the local minimum is saved in line 22. As shown in the histogram in Figure 2, by choosing as objectice function the minimization of the total number of required shifts, the solution obtained when applying the Rollout procedure outperforms the solution generated by the construction phase of pure GRASP. These results have guided us to propose a brand new hybrid algorithm, where the Rollout procedure is used in place of the construction phase of GRASP. We call this procedure hybrid GRASP with Rollout for BDSP.

IV Hybrid GRASP with Rollout

In this innovative approach, we have designed an hybrid GRASP that uses as construction procedure the Rollout algorithm described in Section A.

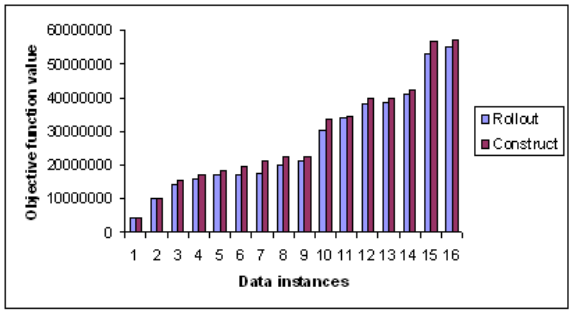

Figure 2: Rollout solution is always better than GRASP construction solution.

At each iteration, a feasible bus driver schedule is constructed by applying the Rollout procedure and, starting from the computed solution, the local search phase described in Section B is applied. The scope of the local search is to improve the schedule exchanging partial shifts and iteratively repeated until we obtain a schedule where there is no interchange that leads to an improvement.

A Computational Results

In our computational experiments we have chosen as objective function the minimization of the total number of required shifts. Our aim here is to evaluate the effectiveness of the GRASP algorithm when combined with Rollout. Computational results obtained with two heuristics Rollout and GRASP using Rollout in the construction phase of the initial solution are reported. All numerical tests were carried out on a Pentium 4 with CPU 3.20 Ghz and 1.00 Gb of memory. The code has been written in C and compiled with DEV-C++[2] version 5.0 beta 9.2. To link our investigation to the real world, data provided by PluService Srl of Senigallia[3] have been used. These data provided by PluService Srl have been utilized to compare the two techniques described above and to verify their effectiveness. Furthermore, their experience in the field of public transportation has allowed us to specify in great detail the set of rules that regulates, the construction of running boards, and the building of shifts. Table 1 and Table 2 report the results obtained with three different procedures (Rollout, GRASP, and GRASP with Roll-

[2] http://www.bloodshed.net/dev/index.html.

[3] Leading company in integrated information systems for local transportation companies with great number of clients in Italy.

out) on 16 Bus Driver Scheduling real-world instances. For each instance, 1000 iterations of the GRASP and GRASP with Rollout have been executed. For each test problem, the first column in Table 1 contains the total number of Pieces-Of-Work, while the last column reports a measure (ϵ) of the relative goodness in terms of objective function value between the solution GR obtained applying GRASP with Rollout and the solution G obtained applying GRASP[4]. The remaining columns of Table 1 provide the values of the objective function obtained by using the tested different procedures. For each test problem, Table 2 reports running time of all proposed methods. From the data reported in Table 1 and Table 2, we observe that the incorporation of Rollout in GRASP was beneficial, improving some of the solutions, but with additional computational burden. In the next future, we are planning to investigate on the quality of solutions that the algorithms will be able to determine, given a fixed amount of computing time.

Table 1: Computational results obtained on real-world BDSP problems by Rollout and GRASP combined with Rollout. R, G, and GR represent the objective function value of the solution found by Rollout, pure GRASP, and GRASP with Rollout, respectively.

Pr./POWs	R	G	GR	ϵ
38	11001913	9001735	8001760	-0.1249
55	8154128	7564132	7512075	-0.0069
71	13002475	9112449	9032354	-0.0088
74	17152279	14716175	15312178	0.0389
76	20108551	15450037	15628071	0.0113
84	22004680	14042163	15022202	0.0652
84	16002785	14002778	14038727	0.0025
114	16173713	13701807	14457704	0.0522
119	16289307	15537121	15640948	0.0066
142	34026490	31192465	29106375	-0.0716
162	32007135	27067061	29006866	0.0668
168	40008713	32084677	31008319	-0.0347
173	38137340	29212896	29658718	0.0150
175	36239686	32677723	33259456	0.0174
197	57010123	45127201	45113061	-0.0003
250	54011482	41151322	43017125	0.0433

In all cases the objective function value obtained applying GRASP with Rollout is better than the one obtained using Rollout. The results show, that for 6 cases out of a total of 16, the solution found by the new version (i.e., GRASP with Rollout) is better than the solution obtained by GRASP. As far as the results obtained with GRASP are concerned, we note that the procedures used in this section are better in terms of the objective function value, but we have a worsening in terms of time.

[4] $\epsilon = \frac{GR-G}{GR}$. Note that $\epsilon = 0$ if the two algorithms find the same objective function value solution.

Table 2: Computational experience on real-world BDSP problems. $Rsec$, $Gsec$, and $GRsec$ are the number of seconds needed to find the solution by Rollout, pure GRASP, and GRASP with Rollout, respectively.

Pr./POWs	R	G	GR
38	0	11	512
55	0	14	1093
71	2	37	3234
74	2	38	3713
76	1	32	2168
84	9	55	9395
84	2	39	435
114	7	55	11300
119	3	75	3556
142	24	225	30719
162	23	185	27070
168	38	311	42210
173	31	312	103305
175	15	145	18301
197	82	350	165796
250	67	494	160606

V Conclusions and Future Works

In this paper, we have analyzed the Bus Driver Scheduling Problem that is an important aspect of the Transportation Planning System. We used the recently proposed procedure Rollout algorithm, and we have shown that it is always better than the construction phase of pure GRASP. The good computational results lead us to devise a brand new hybrid meta-heuristic: GRASP with Rollout. From the description of Rollout, it is evident that the computational cost of each iteration depends on the number of times the base heuristic is applied. A decrease of the computation cost in the Rollout algorithm could be achieved using approximation functions. We plan in the feature to experiment with this approach as will combine Rollout with other modern and efficient meta-heuristics, such as VNS (Variable Neighborhood Search) and PR (Path-Relinking).

References

[1] D. P. Bertsekas, "Differential training of rollout policies," tech. rep., Apperas in Proc. of the 35th Allerton Conference on Communication, Control, and Computing, Allerton Park, Ill., 1997.

[2] A. Wren, "Scheduling Vehicles and their Drivers-Forty Years' Experience," Tech. Rep. 2004.03, University of Leeds, School of Computing Research, 2004.

[3] G. Desaulniers and M. D. Hickman, "Public Transit," Tech. Rep. G-2003-77, Les Cahiers du GERAD, 2003.

[4] A. Wren and J. M. Rousseau, "Bus Driver Scheduling Problem-An Overview," in *Computer-Aided Transit Scheduling* (I. B. J. R. Daduna and J. R. Paixão, eds.), pp. 173–187, Lecture Notes in Economics and Mathematical Systems 430, Springer-Verlag, Heidelberg, 1995.

[5] A. Wren, *Computer Scheduling of Public Transport*. North-Holland, Amsterdam, 1981.

[6] J. M. Rousseau, *Computer Scheduling of Public Transport 2*. North-Holland, Amsterdam, 1985.

[7] J. R. Daduna and A. Wren, *Computer-Aided Transit Scheduling*. Lecture Notes in Economics and Mathematical Systems 308, Springer-Verlag, Heidelberg, 1988.

[8] M. Desrochers and J. M. Rousseau, *Computer-Aided Transit Scheduling*. Lecture Notes in Economics and Mathematical Systems 386, Springer-Verlag, Heidelberg, 1992.

[9] J. R. Daduna, I. Branco, and J. M. P. Paixão, *Computer-Aided Transit Scheduling*. Lecture Notes in Economics and Mathematical Systems 430, Springer-Verlag, Heidelberg, 1995.

[10] N. H. M. Wilson, *Computer-Aided Transit Scheduling*. Lecture Notes in Economics and Mathematical Systems 471, Springer-Verlag, Heidelberg, 1999.

[11] S. Voß and J. Daduna, *Computer-Aided Transit Scheduling*. Lecture Notes in Economics and Mathematical Systems 505, Springer-Verlag, Heidelberg, 2001.

[12] E. Marchitto, *The Bus Driver Scheduling Problem: a new mathematical model and several alternative meta-heuristic algorithms*. PhD thesis, University of Rome "La Sapienza", Dipartimento di Statistica, Probabilità e Statistiche Applicate, 2006.

[13] R. De Leone, P. Festa, and E. Marchitto, "The Bus Driver Scheduling Problem: a new mathematical model and a GRASP approximate solution," Tech. Rep. 22, University of Napoli, Federico II, May 2006.

[14] G. Tesauro and G. R. Galperin, "On-line policy improvement using monte carlo search, presented at the 1996 neural information processing systems conference, denver, co," in *Advances in Neural Information Processing Systems 9* (M. M. et al., ed.), MIT Press, 1997.

[15] D. P. Bertsekas, J. N. Tsitsiklis, and C. Wu, "Rollout algorithms for combinatorial optimization problems," *Journal of Heuristics*, vol. 3, pp. 245–262, 1997.

[16] D. P. Bertsekas, "Rollout algorithms for constrained dynamic programming," tech. rep., Report LIDS 2646, 2005.

[17] D. P. Bertsekas and D. A. Castanon, "Rollout algorithms for stochastic scheduling problems," *Journal of Heuristics*, vol. 5, pp. 89–108, 1999.

[18] F. Guerriero and M. Mancini, "Parallelization strategies for rollout algorithms," *Computational Optimization and Applications*, vol. 31, pp. 221–244, 2005.

[19] F. Guerriero, M. Mancini, and R. Musmanno, "New rollout algorithms for combinatorial optimization problems," *Optimization Methods and Software*, vol. 17, pp. 627–654, 2002.