UCT method in stochastic transportation problems

Jacek Mańdziuk
Warsaw University of Technology
Koszykowa 75, 00-662 Warsaw, Poland
mandziuk@mini.pw.edu.pl

Maciej Świechowski
Systems Research Institute P.A.S.
Newelska 6, 01-447 Warsaw, Poland
m.swiechowski@ibspan.waw.pl

Abstract

Capacitated Vehicle Routing Problem (CVRP) is a well-known NP-hard optimization problem. In this paper, a dynamic version of the CVRP is considered which takes into account traffic jams (TJ). TJ occur randomly according to pre-defined intensity and length distributions. In effect, the static VRP is transformed into a non-deterministic scheduling problem with high uncertainty factor and variable, changing in time internal problem parameters.

Our proposed solution to CVRP with TJ (CVRPwTJ) relies on the UCT (Upper Confidence Bounds applied to Trees) method. UCT is a simulation-based algorithm and currently a state-of-the-art approach to some popular games, which are intractable in the classical ways (rooted in the alpha-beta type search). In short, UCT is an extension of the Monte Carlo Tree Search (MCTS) method, however, unlike MCTS which makes use of uniformly distributed simulations in a game tree (in order to find the most promising move), the UCT algorithm aims at maintaining an optimal balance between exploration and exploitation which results in more frequent visits to and deeper expansion of the most promising branches of a game tree.

The paper is the first attempt of applying the UCT algorithm in the domain of dynamic transportation problems. The most challenging issue here is finding a suitable mapping of the CVRPwTJ onto a tree-like problem representation required by the UCT. Furthermore, in order to prevent the size of the tree from explosive growth, an efficient mechanism for child nodes selection must be applied.

Our approach is compared with two heuristic methods, which rely on 2-OPT-type local route optimizations intended to avoid traversing jammed edges if a low-cost local reordering of clients is possible. Initial results are very promising and give hope for wider applicability of the proposed solution in the domain of dynamic optimization problems.

1 Introduction

Vehicle Routing Problem [Dantzig and Ramser, 1959] along with its numerous variants (e.g. Capacitated Vehicle Routing Problem) is a widely known combinatorial optimization task. Its practical (industrial) background and relevance provokes strong ongoing interest of Artificial Intelligence (AI) community in finding new efficient methods of its solving despite some already existing and well-established heuristic and approximate ones.

Upper Confidence Bounds applied to Trees (UCT) method [Kocsis *et al.*, 2006; Browne *et al.*, 2012], is currently a state-of-the-art approach to game playing in the case of games for which a compact and easily computable position assessment function is not known.

The main advantage of using UCT in games is its adaptability (to the changing game situation) and long-term reliability of position assessment. An additional asset of UCT is its "knowledge-free" nature [Mańdziuk, 2010] - there is no requirement for domain-specific knowledge, except for the formal game definition, which is indispensable in the move generation process and for detection and evaluation of the final states of the game.

In the view of the above-listed UCT qualities we conducted research on possible ways of incorporating the UCT algorithm, in its basic or modified form, into specific class of Vehicle Routing Problems in which frequent traffic jams may occur while traversing the edges of the problem's solution. In particular, this research aims at verification of the UCT capability to flexibly address the so-called "exploration vs. exploitation dilemma", i.e. the issue of balancing the usage of discovered best solutions vs. finding the new ones with respect to highly variable work conditions. Such a "plasticity" of the solution method seems to be indispensable for efficient solving of the CVRP with Traffic Jams (CVRPwTJ).

It is worth underlying that, to the best of our knowledge, this paper presents the first attempt to solving the CVRPwTJ by means of the UCT method. Therefore, we had to make several decisions related to particular implementation and usage of the method in a given application domain. The main issue was related to the CVRPwTJ problem representation in the form of a graph (which is a desirable representation for a UCT tree-search method), and definition of the set of UCT actions (possible "moves" in a given state of a partial solution) and their interpretation per analogy to game moves. An-other critical decision was introduction of combined actions involving two vehicles and simultaneous modification of their partial solutions / routes.

The rest of the paper is organized as follows: in the next section a formal definition of the CVRPwTJ is provided. Sections 3 and 4, respectively summarize the UCT method and the way we propose to apply it to solving CVRPwTJ. Section 5 is devoted to presentation of an experimental setup, simulation results and their comparison with two heuristic approaches. A summary of the main contribution and directions for future research conclude the paper.

2 Capacitated Vehicle Routing Problem with Traffic Jams

Vehicle Routing Problem was formally formulated in 1959 [Dantzig and Ramser, 1959] and subsequently proved to be NP-hard in 1981 [Lenstra and Kan, 1981]. In short the problem consists in assigning a number of homogenous vehicles to a number of clients, where each client has a certain 2D location and a certain demand of (homogeneous) goods and the goal (optimization objective) is to deliver demanded goods to all clients while minimizing the sum of vehicles routes' costs (lengths). There are a few additional constraints which must be fulfilled, i.e. each client must be served by exactly one vehicle and each vehicle's route must start and end in the depot (defined by its 2D coordinates).

The basic formulation of VRP does not impose any limit on the number of clients that can be served by a given vehicle. For practical reasons, however, the upper limit on vehicles' capacity is often imposed, leading to the Capacitated Vehicle Routing Problem (CVRP) definition.

Since VRP/CVRP is NP-Hard, no polynomial method of solving the problem is known and perfect solutions can only be obtained for relatively small-size problems. For the real-life problem instances approximation algorithms must be applied. Among the exact algorithms, one can distinguish the following three main approaches: full tree search (e.g. spanning tree and shortest path relaxations method [Christofides *et al.*, 1981]), dynamic programming (e.g. [Eilon *et al.*, 1976] in the case of problems with an a priori known number of required vehicles) and integer programming (e.g. three-index vehicle flow formulation [Fisher and Jaikumar, 1978]). There are multiple approximation algorithms for VRP/CVRP, most of them designed to address specific problem formulations. For example, Savings algorithm [Clarke and Wright, 1964] assumes that the number of vehicles is not limited. Other well-known methods include Multi-route improvement algorithm [Breedam, 1994], Sweep algorithm [Gillett and Miller, 1974], Ant Colony optimization [Dorigo, 1992] or Particle Swarm Optimization [Khouadjia *et al.*, 2010; Okulewicz and Mańdziuk, 2013; 2014].

The variant of CVRP considered in this paper differs significantly from the above-mentioned static versions by introducing a high degree of uncertainty by means of traffic jams, which may dynamically occur on the particular edges (atomic parts) of the planned vehicles' routes. The existence of a traffic jam increases the cost of traversing a certain edge, usually to the extent that requires some re-modeling of the currently planned route. In the proposed solution, these dynamic changes are handled on-line by appropriate actions taken to alleviate their impact (see section 5.3 for the details). Generally speaking, the times of occurrences, intensities and time spans of the traffic jams are generated according to some probability distributions whose parameters (but not the actual realizations) are pre-defined (see section 5.1).

3 Upper Confidence Bounds Applied to Trees

UCT is a simulation-based algorithm, which proved to be successful mainly in multi-step decision-making (acting) under uncertainty, in particular in the case of building playing agents for several demanding games, such as Go [Gelly and Silver, 2011; Browne *et al.*, 2012] or Havannah [Teytaud and Teytaud, 2010]. The UCT is also a state-of-the-art approach to the so-called General Game Playing [Genesereth *et al.*, 2005; Świechowski and Mańdziuk, 2014; Walędzik and Mańdziuk, 2014].

The method consists in performing multiple simulations of possible continuations of the game from the current state. Instead of performing fully random rollouts, as is the case of the MCTS method, it proposes a more selective approach to choosing continuations worth analyzing, making the obtained results more meaningful. In each state, UCT advises to first try each action once and then, whenever the same position is analyzed again, choose move a* according to the following formula:

$$a^* = arg \max_{a \in A(s)} \left\{ Q(s,a) + C\sqrt{\frac{ln\left[N(s)\right]}{N(s,a)}} \right\} \tag{1}$$

where $A(s)$ is a set of all actions available in state s, $Q(s,a)$ denotes the average result of playing action a in state s in the simulations performed so far, $N(s)$ - a number of times state s has been visited and $N(s,a)$ - a number of times action a has been sampled in this state. Constant C controls the balance between exploration and exploitation, since the formula postulates choosing actions with the highest expected rewards and, at the same time, avoiding repetitive sampling of the same actions while others might yet prove more beneficial.

Direct implementation of the approach described above is, however, impossible due to memory limitations. With sufficient time it would lead directly to storing the whole game tree in memory, which is unfeasible, except for the simplest games. Therefore, each simulation (rollout) actually consists of two phases: a strict UCT phase and a Monte-Carlo (MC) phase, the latter one consisting in truly random rollouts. In-memory game tree is built iteratively and in each internal simulation the UCT strategy, i.e. formula (1) is applied only until the first not-yet-stored game state is reached. It is, then, added to the in-memory game tree and strictly random MC rollout is performed further down the tree (see Figure 1, reprinted from [Chaslot *et al.*, 2008]).

With the UCT simulations finished in a given time step, choosing the next move is simply a matter of finding the action with the highest $Q(s,a)$ value in the current state.

Recently, UCT gained attention in the area of probabilistic planning implemented by the Markov Decision Process

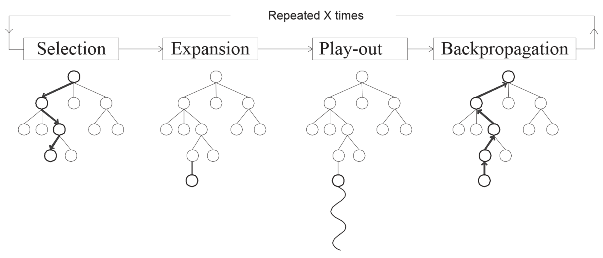

Figure 1: Operational scheme of the UCT method. First a UCT tree is traversed until a leaf node is reached (Selection), then the tree is expanded by adding a new leaf (Expansion), next a random rollout is performed until the end-of-game state (Play-out) and finally the result is propagated back in the UCT tree (including the newly added leaf). This scheme is repeated a given number of times. Each such realization is called and internal UCT iteration.

(MDP) model [Kolobov *et al.*, 2012; Keller and Eyerich, 2012; Feldman and Domshlak, 2014]. Successful application of UCT in this area motivated our work on CVRPwTJ.

4 UCT in CVRP with Traffic Jams

The UCT method, due to its generality and "knowledge-free" nature [Mańdziuk, 2010], seems to be a good candidate for a general-purpose widely-applicable metaheuristic in the domain of decision-making problems if only they can be transformed into tree-based representation. In this section the proposed way of applying the UCT method to the CVRP with Traffic Jams is presented. Before description of the method let us present the way the static CVRP benchmarks are transformed into stochastic CVRPwTJ instances.

For a given benchmark problem the solution search process is divided into a number of discrete time steps and in each of them all active trucks (i.e. those which left the depot) move to their next clients. The process ends when all vehicles move back to the depot after having visited all the customers. The initial conditions (i.e. the number of available trucks, their capacity, clients' requests and the coordinates of the depot and the clients) are given in the benchmark set definition. What makes the CVRPwTJ different from the above (static) CVRP definition is considering road conditions (traffic jams). Specifically, at each time step, for each edge (a direct link between two clients or a client and a depot) the TJ is imposed with probability P. If TJ happens to appear on a given edge a it is assigned a randomly selected intensity $I(a)$ and length $L(a)$ (measured in time steps).

Therefore, for each edge a of cost $c(a)$, if a traffic jam $TJ(a)$ with intensity $I(a)$ and length $L(a)$ appears on that edge, its current cost is modified to $c(a) \cdot I(a)$ for the next $L(a)$ time steps and reduced to the previous value $c(a)$, afterwards. If a TJ happens to be selected for an edge a which is currently jammed (was jammed k time steps before, for some k, with the TJ length $L(a) > k$ and intensity $I(a)$), then the length of the TJ on that edge is increased by the newly selected TJ' length, but the cost of that edge is not increased - it remains at the level of $c(a) \cdot I(a)$ (though, for a longer time period). This way we avoid TJ intensity multiplications which might otherwise have easily led to explosive growth.

The above TJ assignment (with probability P) is applied at the beginning of each time step and for all edges. Since at time zero there are no TJ imposed yet, we start our algorithm by finding an initial solution to the static version of the CVRP problem.

4.1 Initial Solution

The initial solution, in the form of a set of vehicles' routes is obtained **for the static problem instance** with the help of a modified Clark and Wright [Clarke and Wright, 1964] savings algorithm [Pichpibul and Kawtummachai, 2012]. Each route commences and ends in a depot and the routes are pairwise separated (except for the initial and final position which is always a depot). This set of optimized paths, with depot being the first and the last element, forms the input to the proposed UCT approach.

4.2 UCT Forest

Suppose the initial solution is composed of k routes, i.e. uses k trucks. Then, as stated above, the initial UCT tree is actually a forest composed of k trees - each in the form of a path with the first and the last elements being a depot. The consecutive elements on each path denote the clients visited by respective trucks in subsequent time steps. For example, the third elements in all paths represent the set of clients visited in the second step of the solution.

The internal UCT simulations are performed at each time step simultaneously from all root nodes of all k trees. As explained in section 3 the trees are gradually extended (one leaf node at each simulation), though there are two main differences compared to the "classical" UCT usage.

First of all, the next compound step (movement of all k trucks) is a result of a combined knowledge obtained from all k trees. More precisely, in each tree the most promising action is selected, then these k selected actions are sorted in descending order based on their UCT values (i.e values $C\sqrt{\frac{ln[N(s)]}{N(s,a)}} - Q(s,a)$ in (2)) and afterwards executed in this order. Please note, that execution of an action in one tree may disable some further actions (in subsequent trees). In that case the next best action in the latter tree is selected instead.

Second of all, since the shorter the solution the better, the UCT formula (1) is modified to the following version (2), which favors the shorter average outcomes $Q(s,a)$:

$$a^* = arg \max_{a \in A(s)} \left\{ C\sqrt{\frac{ln\,[N(s)]}{N(s,a)}} - Q(s,a) \right\} \quad (2)$$

Once the simulation is completed a compound result from all k trees (the sum of routes of k trucks) is back-propagated to the root nodes of these k trees.

After a certain number of internal simulations the actual (real) decision regarding the movement of the k trucks is made according to the smallest $Q(s,a)$ value among the child nodes in each of the k trees. These $Q(s,a)$ values are sorted in an ascending order (i.e. first the action in the tree with the lowest $Q(s,a)$ is executed, then the action in the tree with the second-lowest $Q(s,a)$, etc.).

Table 1: Actions *level-0* and *level-1*. **Ac.** denotes the code of an action, **Lev.** is level-type, **Conditions** - is the set of prerequisites and **Action** is the actual action taken. $+TJ$ / $-TJ$ denote the fact that the edge currently planned to be traversed is jammed / not jammed, respectively.

Ac.	Lev.	Conditions	Action
$A0$	0	$-TJ$	Continue the planned (non-jammed) route.
$A1$	0	$+TJ$	Continue the planned (jammed) route.
$A2$	1	$+TJ$; New route is not jammed.	Move the current client at the end of a route (just before returning to the depot).
$A3$	1	$+TJ$	Move the current client into locally optimal place in a route - see the main text.
$A4$	1	$+TJ$	Insert the first found client to whom there is no TJ before the current client (as the first one).
$A5$	1	$-TJ$; The route is not the one originally planned (has been changed already).	Insert the client to whom the edge from the current state is the cheapest as the current/first one.
$A6$	1	$+TJ$; After reversal the route does not begin with a TJ.	Reverse the route (except for the depot which remains the closing element).

4.3 Possible Actions in the UCT Trees

As stated above, at each step of both the UCT simulations as well as real decisions regarding the vehicles' tours, all possible actions are considered in each of the k root nodes. These actions take into account whether or not the edge currently planned to be traversed is jammed and if so what is the intensity of the TJ. In particular there are three types of actions differing by their complexity: *level-0*, *level-1* and *level-2*, which modify 0, 1 and 2 routes, respectively. In each case, there are some initial pre-conditions under which a given action is legal. Otherwise it is illegal and hence not considered in a given state. The set of actions *level-0* and *level-1* used in our approach is presented in Table 1. All these actions are relatively simple and self-explained. Please only note that in $A3$, for the current client X we find a new location between clients B and C, so as to minimize $|BX| + |XC| - |BC|$.

Except for the above-mentioned 6 actions there are also 3 more complex, *level-2* ones denoted $A7$, $A8$ and $A9$. In the case of $A7$ the current route is finished (by immediately moving to a depot) and a new one is commenced from the depot. The initial (legality) conditions are the following:

- from the current vehicle location there are TJ to all other customers planned for this route;
- the edge from the vehicle location to the depot is not jammed;
- the edge from the depot to the first customer is not jammed.

The two remaining actions $A8$ and $A9$ exchange the customers between the two routes. In $A8$, the replacement is the smallest possible (the exchange is finished with the first non-jammed situation). In $A9$ the exchange is more complex and aims at exchanging the whole parts of the routes (to some extent similarly to the crossover operation in Genetic Algorithms). In both cases the following prerequisites must be fulfilled:

- one of the routes begins with a traffic jam;
- after exchange both routes do not begin with traffic jams;
- after exchange capacity constraint is fulfilled for both vehicles.

All the above-mentioned actions are based on the following underlying rationale: if the currently selected candidate edge is not jammed then traverse it, otherwise try to enhance the planned route (by avoiding the traffic jam) by means of local changes in the planned orders of visiting clients. In the case of actions $A0$-$A6$ the in-route optimization takes place. Action $A5$ is the only one which allows to optimize a route which is not jammed. It may be particularly suitable when the route is far from optimal due to some changes forced in earlier steps. Action $A7$ immediately completes the current route (since there is neither TJ on the edge leading to the depot nor on the one from the depot to the next planned customer, the next turn will potentially be made on an non-jammed edge starting from the depot). Finally, actions $A8$ and $A9$ exchange customers between the two routes so as to reach locally (temporarily) non-jammed solution. In theory, one may proceed with defining more complex and more sophisticated actions, e.g. the ones involving three or more routes (trucks), but such approach immediately becomes infeasible due to computational complexity reasons.

5 Experimental Results

In this section, the experimental setup and traffic jams' parameterization are presented along with the results of applying the proposed approach to a set of popular static CVRP benchmarks modified by imposing the TJ. The results are compared with the static solution (not reacting to TJ occurrence) and two local-search-based heuristic methods.

5.1 Experimental Setup

In the experiments, the set of static benchmark problems specified in Table 2 was used. The benchmarks were downloaded from the CVRP webpage [NEO. Networking and Emerging Optmization, 2013] and modified into CVRPwTJ according to the following TJ uniform probability distributions:

$P \in \{0.02; 0.05; 0.15\}$
$I = U_{INT}[10, 20]$
$L = U_{INT}[2, 5]$

where $U_{INT}[a, b]$ denotes random uniform selection of any integer x such that $a \leq x \leq b$. Based on the initial tests the value of C in (2) was set to the length of the initial solution found for the static instance.

A solution to a static version of the P-n101-k4 benchmark is presented in Figure 2.

5.2 Two Heuristical Approaches

The results of proposed method were compared with the static solution (obtained with the improved Clark and Wright algorithm [Pichpibul and Kawtummachai, 2012] mentioned in section 4.1) applied to the dynamic benchmark instance and

Table 2: Static benchmark instances used as a base for defining CVRPwTJ instances. Columns, from left to right, denote respectively: the name of a benchmark set, number of clients, number of vehicles, truck's capacity, the best (static) solution rounded to the integer value and the number of UCT internal simulations per move (performed action) used in the experiments.

Instance	#C	#V	Cap.	$BEST$	#Sim.
P-n19-k2	19	2	160	212	80 000
P-n45-k5	45	5	150	510	26 000
A-n54-k7	54	7	100	1167	26 000
A-n69-k9	69	9	100	1168	26 000
A-n80-k10	80	10	100	1764	4 000
P-n101-k4	101	4	400	681	4 000

with two heuristic algorithms which use the following local optimization mechanisms: **(H1)** and **(H2)**, respectively.

(H1): In this heuristic the initial solution is traversed step by step until the TJ occurs on the current edge. In that case a local 2-OPT optimization improvement is performed and compared with the cost of traversing the jammed edge.

More precisely, let's denote by $d(x, y)$ the length of a direct edge from location x to location y and suppose that the customers (nodes) in the initial solution are numbered consecutively from 0 (depot) to n, i.e. the initial route is then of the form $0 - 1 - 2 - \ldots - n - 0$. Let's now assume that after the first step (moving from 0 to 1) a TJ on the edge $(1, 2)$ is encountered. In such a case, the node

$$p = arg \min_{p} \{d(1, p) + d(2, p + 1)\} \quad (3)$$

is found and an alternative route omitting the edge $(1, 2)$ is considered (see Figure 3). The new route is accepted if

$$d(1, p) + d(2, p + 1) < d(1, 2) + d((p, p + 1) \quad (4)$$

Otherwise a jammed edge $(1, 2)$ is traversed in the next step.
(H2): In this heuristic, similarly to H1, the initial solution is traversed step by step until a TJ occurs on the current edge. Let's again assume that there is a TJ on edge $(1, 2)$. In such a case H2 tries to omit node 2, i.e. proceeds along $(1, 3)$ - if not jammed - and gets back to node 2 at the earliest possible occasion in the next steps. If $(1, 3)$ is also jammed the edge $(1, 4)$ is tried and both 2 and 3 are placed in a waiting queue, etc.

In effect, the omitted customers are placed in a waiting line and in the next steps, before proceeding along the current route H2 verifies if any of these omitted clients can be served in this step without traversing a jammed edge. The waiting customers are tried out in the order of their appearance in the queue (from the latest to the newest).

5.3 Results

For each pair (instance, P) 50 experiments were performed for each of the compared approaches (static, H1, H2 and UCT). Please note, that for a given experimental trial (one out of 50) the same TJ realizations were used in all approaches. Clearly these realizations differed across these 50 trials. It should be noted that these TJ realizations were not known to the tested algorithms beforehand (i.e. unless the traffic jams actually materialized in a given time step). Hence, solving the CVRPwTJ requires truly *on-line* and self-adaptive probabilistic planning capabilities. The results are presented in Tables 3 and 4.

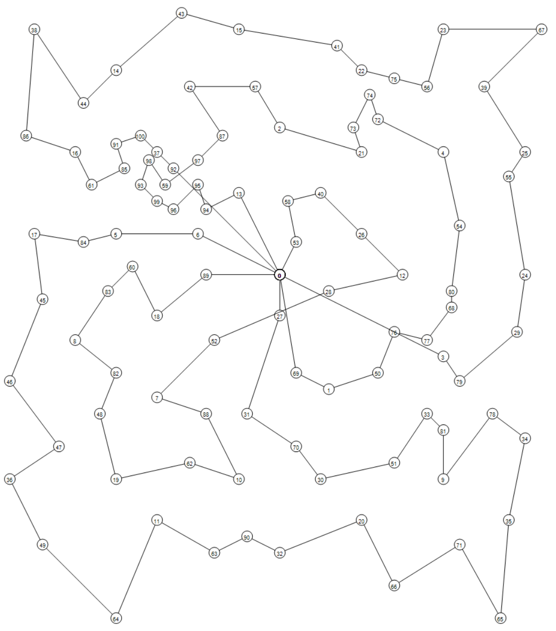
Figure 2: Solution of the P-n101-k4 benchmark problem.

On a general note, it can be seen that UCT is superior to other approaches, whereas the static solution is clearly the worse option. Furthermore, it can be observed that the UCT solutions are not only the shortest among the competing approaches but also have the lowest standard deviations. Additionally, the superiority of UCT is prevailing when the TJ probability increases.

The relatively weaker results are obtained in the case of bigger-size benchmarks with low P. The reasons for that situation are two-fold: first of all, the number of internal UCT simulations seems to be too small for the P101 problem instance. Secondly, in the case of big benchmark set with low probability of TJ the quality of initial solution (which is relatively very high in the case of method used [Pichpibul and Kawtummachai, 2012]) plays a crucial role in the final result (due to a low degree of problem's dynamism). In the case of efficient initial solution and relatively stable problem instance local 2-OPT type optimization heuristic clearly constitute a strong, competitive approach. We believe, however, that with a higher number of simulations, UCT would further improve its performance and possibly outperform both H1 and H2. Verification of this hypothesis is one of our current research goals.

The other situation in which UCT did not show its advan-

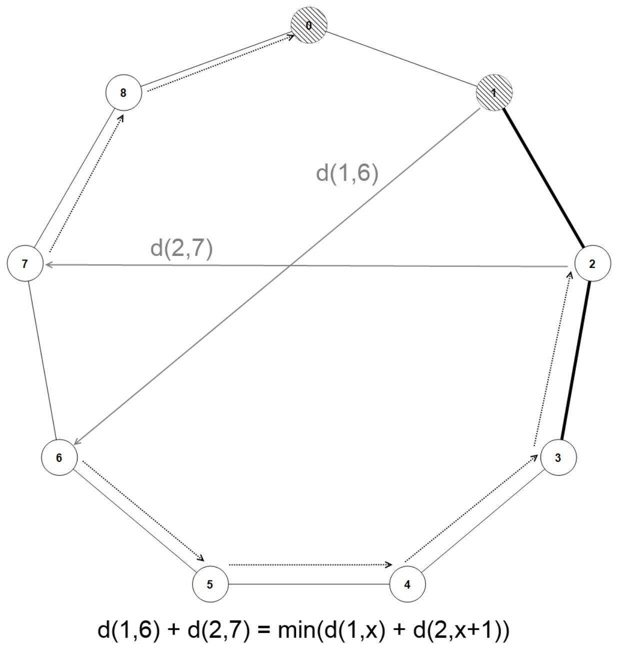

Figure 3: 2-OPT local optimization used by heuristic H1. See description within the text.

Table 3: The numbers of trials in which a given approach appeared to be the best one. Please note that due to ties, the sums in rows may exceed 50. The winning algorithm for each pair (instance, P) is bolded.

Instance	P	**Static**	**H1**	**H2**	**UCT**
P19	0.02	19	37	**46**	45
	0.05	9	27	34	**38**
	0.15	1	5	15	**37**
P45	0.02	5	23	26	**28**
	0.05	1	9	13	**38**
	0.15	0	0	0	**50**
A54	0.02	1	11	20	**35**
	0.05	0	2	9	**39**
	0.15	0	0	0	**50**
A69	0.02	0	11	20	**31**
	0.05	0	3	5	**42**
	0.15	0	0	0	**50**
A80	0.02	0	13	17	**27**
	0.05	0	1	10	**39**
	0.15	0	0	0	**50**
P101	0.02	0	**25**	24	10
	0.05	0	12	**21**	20
	0.15	0	0	6	**44**

Table 4: The average values and standard deviations (in parentheses) across 50 trials. The lowest (best) results are bolded.

Instance	P	**Static** (σ)	**H1** (σ)	**H2** (σ)	**UCT** (σ)
P19	0.02	412.4 (204.5)	287.2 (84.6)	269.0 (62.7)	**266.9 (51.6)**
	0.05	582.9 (258.8)	345.2 (126.8)	307.34 (92.4)	**283.9 (62.0)**
	0.15	1291.0 (473.9)	817.9 (403.9)	587.5 (247.3)	**451.6 (148.9)**
P45	0.02	1016.1 (309.2)	736.6 (207.0)	667.6 (147.6)	**607.1 (29.9)**
	0.05	1372.6 (375.4)	835.1 (221.9)	739.9 (144.1)	**623.2 (38.4)**
	0.15	3650.4 (553.2)	2256.7 (544.9)	1597.5 (402.0)	**800.1 (125.7)**
A54	0.02	2006.6 (519.7)	1626.2 (483.0)	1441.4 (345.7)	**1265.3 (66.7)**
	0.05	3218.6 (928.8)	2052.2 (422.9)	1798.2 (369.8)	**1401.8 (125.8)**
	0.15	6793.7 (1518.6)	4181.8 (888.4)	3253.4 (823.4)	**1738.2 (220.8)**
A69	0.02	2144.5 (490.3)	1646.8 (406.4)	1505.9 (373.7)	**1284.0 (52.1)**
	0.05	3275.9 (791.6)	2018.1 (415.1)	1849.3 (423.9)	**1436.6 (136.6)**
	0.15	6937.0 (1048.7)	4401.1 (945.8)	3394.4 (806.2)	**1815.8 (222.9)**
A80	0.02	2937.6 (689.7)	2231.1 (357.8)	2151.9 (307.2)	**1967.0 (86.1)**
	0.05	4434.0 (976.7)	2865.7 (498.6)	2543.6 (391.4)	**2137.1 (133.4)**
	0.15	9073 (1527.8)	5765.3 (851.4)	4549.0 (816.7)	**2594.3 (218.3)**
P101	0.02	1538 (283.6)	861.1 (85.9)	**848** (84.3)	863.2 (**51**)
	0.05	2337 (439.4)	1035.3 (156.0)	972.3 (128.4)	**963.7 (73.8)**
	0.15	6465 (609.9)	2750.9 (689.6)	1758.0 (319.9)	**1379.2 (209.9)**

tage is the case of the easiest problem P19 and low probability of TJ. This problem is easy enough to be solved by local optimization methods and therefore all H1, H2 and UCT performed comparably well for this benchmark. Please note, however, that all of them showed significant improvement over the static solution.

6 Conclusions and Directions for Future Research

In this paper a new approach to the Capacitated Vehicle Routing Problem with Traffic Jams is presented. The solution is based on a UCT algorithm which was hitherto applied mainly in game domain (in particular to games for which a compact and meaningful evaluation function is not known) and to MDP-based probabilistic planning. Application of the UCT method to the CVRPwTJ required suitable problem representation (in the form of a forest of trees) and specific definition of legal actions to be performed in these trees in order to prevent the method from explosive growth of memory requirements.

The UCT-based results were successfully compared with two heuristic methods, locally optimizing the routes in case of a traffic jam occurrence. The advantage of our approach gives hope for its further successful development and usefulness in the domain of transportation problems.

One of the advantages of proposed approach is the possibility to use any optimization method for finding the initial solution of the static problem. Our choice was a parallel version of the improved savings method [Pichpibul and Kawtummachai, 2012], which proved to be efficient, but should other methods seemed to be more appropriate there would be no problem in their application.

Our current focus is on verification of the potential UCT results' improvement with the increased number of internal simulations in the case of $P-n101-k4$ benchmark set (due to memory limitations the number of UCT internal simulations was relatively low for this problem instance). Furthermore, we plan to compare our method with appropriately designed and tuned Ant Colony System (ACS) and Particle Swarm Optimization (PSO) approaches. Our initial comparison with the ACS showed an upper hand of the UCT solutions, however more experimental evaluations are necessary to confirm these preliminary conclusions.

Acknowledgments

The research was financed by the National Science Centre in Poland grant number DEC-2012/07/B/ST6/01527.

References

[Breedam, 1994] A. Van Breedam. *An analysis of the behavior of heuristics for the vehicle routing problem for a selection of problems with vehicle-related, customer-related, and time-related constraints*. PhD thesis, University of Antwerp, Belgium, 1994.

[Browne *et al.*, 2012] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[Chaslot *et al.*, 2008] Guillaume Chaslot, Mark H. M. Winands, Istvan Szita, and H. Jaap van den Herik. Cross-Entropy for Monte-Carlo Tree Search. *ICGA Journal*, 31(3):145–156, 2008.

[Christofides *et al.*, 1981] N. Christofides, A. Mingozz, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–282, 1981.

[Clarke and Wright, 1964] G. Clarke and J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[Dantzig and Ramser, 1959] G. B. Dantzig and J. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, oct 1959.

[Dorigo, 1992] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, 1992.

[Eilon *et al.*, 1976] S. Eilon, C. Watson-Gandy, and N. Christofides. *Distribution Management: Mathematical Modelling and Practical Analysis*. Griffin, first edition, jan 1976.

[Feldman and Domshlak, 2014] Zohar Feldman and Carmel Domshlak. On monte-carlo tree search: To mc or to dp? In *Proceedings of ECAI-14. 21st European Conference on Artificial Intelligence*, 2014.

[Fisher and Jaikumar, 1978] M. Fisher and R. Jaikumar. *A Decomposition Algorithm for Large-scale Vehicle Routing*. Paper / Department of Decision Sciences, Wharton School, University of Pennsylvania. Philadelphia, Pa. Dep. of Decision Sciences, Wharton School, Univ. of Pennsylvania, 1978.

[Gelly and Silver, 2011] S. Gelly and D. Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

[Genesereth *et al.*, 2005] M. R. Genesereth, N. Love, and B. Pell. General game playing: Overview of the aaai competition. *AI Magazine*, 26(2):62–72, 2005.

[Gillett and Miller, 1974] B. Gillett and L. Miller. A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 22(2):340–349, 1974.

[Keller and Eyerich, 2012] Thomas Keller and Patrick Eyerich. Prost: Probabilistic planning based on uct. In *Proceedings of International Conference on Automated Planning and Scheduling*, 2012.

[Khouadjia *et al.*, 2010] M. Khouadjia, E. Alba, L. Jourdan, and E-G. Talbii. Multi-Swarm Optimization for Dynamic Combinatorial Problems: A Case Study on Dynamic Vehicle Routing Problem. In *Swarm Intelligence*, volume 6234 of *Lecture Notes in Computer Science*, pages 227–238. Springer, Berlin / Heidelberg, 2010.

[Kocsis *et al.*, 2006] L. Kocsis, C. Szepesvri, and J. Willemson. Improved monte-carlo search. Working Paper, 2006.

[Kolobov *et al.*, 2012] Andrey Kolobov, Mausam, and Daniel S. Weld. LRTDP versus UCT for online probabilistic planning. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[Lenstra and Kan, 1981] J. K. Lenstra and A.H.G. R. Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11:221–227, 1981.

[Mańdziuk, 2010] J. Mańdziuk. *Knowledge-Free and Learning-Based Methods in Intelligenet Game Playing*, volume 276 of *Studies in Computational Intelligence*. Springer-Verlag, Berlin, Heidelberg, 2010.

[NEO. Networking and Emerging Optmization, 2013] NEO. Networking and Emerging Optmization, 2013. http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/.

[Okulewicz and Mańdziuk, 2013] M. Okulewicz and J. Mańdziuk. Application of Particle Swarm Optimization Algorithm to Dynamic Vehicle Routing Problem. In *Artificial Intelligence and Soft Computing*, volume 7895 of *Lecture Notes in Computer Science*, pages 547–558. Springer Berlin Heidelberg, 2013.

[Okulewicz and Mańdziuk, 2014] M. Okulewicz and J. Mańdziuk. Application of Particle Swarm Optimization Algorithm to Dynamic Vehicle Routing Problem. In *Proceedings of the IEEE Symposium on Computational Intelligence for Human-Like Intelligence*, pages 86–93. IEEE Press, 2014.

[Pichpibul and Kawtummachai, 2012] T. Pichpibul and R. Kawtummachai. An improved clarke and wright savings algorithm for the capacitated vehicle routing problem. *Science Asia*, pages 307–318, 2012.

[Świechowski and Mańdziuk, 2014] M. Świechowski and J. Mańdziuk. Self-adaptation of playing strategies in general game playing. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(4):367–381, 2014.

[Teytaud and Teytaud, 2010] F. Teytaud and O. Teytaud. Creating an upper-confidence-tree program for havannah. *Advances in Computer Games*, pages 65–74, 2010.

[Walędzik and Mańdziuk, 2014] K. Walędzik and J. Mańdziuk. An Automatically-Generated Evaluation Function in General Game Playing. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(3):258–270, 2014.