UCT-Based Approach to Capacitated Vehicle Routing Problem

Jacek Mańdziuk(✉) and Cezary Nejman

Faculty of Mathematics and Information Science,
Warsaw University of Technology,
Koszykowa 75, 00-662 Warsaw, Poland
mandziuk@mini.pw.edu.pl, nejmanc@student.mini.pw.edu.pl

Abstract. Vehicle Routing Problem (VRP) is a popular combinatorial optimization problem which consists in finding an optimal set of routes for a fleet of vehicles in order to serve a specified collection of clients. Capacitated VRP (CVRP) is a version of VRP in which every vehicle has a capacity parameter assigned.

The UCT (Upper Confidence bounds applied to Trees) is a heuristic simulation-based algorithm used for learning an optimal policy in games. The algorithm 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 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 to apply the UCT algorithm to solving CVRP. The critical issue here is suitable mapping of the CVRP onto a game tree structure, which is not straightforward in this problem domain. Furthermore, in order to keep the tree size within reasonable limits the appropriate way of child nodes selection must be considered. Another pertinent issue is interpretation of game-related terms "win" and "loss" in the CVRP context.

Experimental results of several mappings of CVRP to game tree-like structure are presented for a collection of popular benchmark sets.

Keywords: UCT · Routing problems · Dynamic optimization

1 Introduction

Vehicle Routing Problem (VRP) along with its variants is a widely known combinatorial optimization task. Due to its practical relevance there is a strong interest in finding new approaches to solving this problem despite already existing heuristic and approximate methods. UCT method, in turn, is one of the most popular approaches to game playing mainly due to its adaptability and long-term efficiency. An additional asset of UCT is the lack of need for domain-specific knowledge. Taking the qualities of UCT approach and requirements of VRP problem into consideration, we conducted research on possible ways of

L. Rutkowski et al. (Eds.): ICAISC 2015, Part II, LNAI 9120, pp. 679–690, 2015.
DOI: 10.1007/978-3-319-19369-4_60

incorporating the UCT algorithm, in its basic form, into specific class of Capacitated Vehicle Routing Problems (CVRP). The most promising factor of such a combination is the fact that UCT proved to be very effective in solving the so-called "exploration vs. exploitation dilemma", i.e. the issue of balancing the usage of discovered best solutions vs. finding the new ones. This property seems to be very well suited to the nature of VRP/CVRP. Moreover, it is worth noting the novelty of the proposed approach as, to the best of our knowledge, it is the first time the UCT method is applied to solving the VRP/CVRP. The main issues analyzed in this piece of research are related to appropriate problem representation, interpretation of "win" and "loss" situations and finding efficient UCT parameterization. Initial experiments showed that implementation of the basic UCT method with naive problem representation leads to mediocre results. After finding weak points of this preliminary approach the baseline algorithm was modified by raising the importance of exploration as well as discretizing the solution evaluation values. These changes led to the overall improvement of results and allowed making a conclusion that the modified method with discretization factor is a promising way of incorporating UCT into CVRP.

The remainder of the paper is organized as follows: in the next section a formal definition of the CVRP is provided. Section 3 presents the UCT method and the proposed way of its application to solving CVRP. Section 4 is devoted to experimental setup, simulation results and conclusions. The last section summarizes the main contribution of the paper and points directions for future research.

2 CVRP Formulation

VRP was formulated in 1959 [4] and proved to be NP-hard in 1981 [11]. In its base formulation, there is a number of homogenous vehicles and a number of clients (sometimes interpreted as cities) and each client has a certain (known) demand which must be satisfied by (exactly) one of the vehicles. The goal is to deliver demanded goods to all clients while minimizing the sum of vehicles routes' costs (lengths). The delivered goods are homogenous. Each vehicle's route must start and end in the specified depot. Each client as well as the depot has a certain 2-dimensional location. It is worth noting that in the basic formulation of VRP, there is no limit on the number of clients that can be serviced by a single vehicle. In practical applications though, the capacity parameter must be added to a vehicle characteristics in order to ensure that there will be no situation of servicing clients with higher total demand than the vehicle's capacity. Such a formulation leads to CVRP. Since VRP/CVRP is NP-Hard no polynomial method of its solving is known. Thus, exact solutions can only be obtained for small-size problems. In practice, approximation algorithms must be used in order to obtain the results in acceptable time. Among the exact algorithms, there are three main approaches: full tree search, dynamic programming and integer programming. An example of the first approach would be spanning tree and shortest path relaxations method [16]. A dynamic programming method was, for instance, proposed in [6] for problems with known number of vehicles. As for the third approach,

a three-index vehicle flow formulation was presented in [7]. There exist various approximation algorithms for VRP/CVRP, most of them designed to address specific problem formulations, e.g. Savings algorithm [3] which assumes that the number of vehicles is unlimited. Other well-known methods include Multi-route improvement algorithm [1], Sweep algorithm [10], Ant Colony Optimization [5] or Particle Swarm Optimization [12,18].

3 UCT Search Tree Method

UCT (Upper Confidence Bound applied to Trees) is an extension to Monte Carlo Tree Search (MCTS) method developed in 2006 [13]. MCTS is an optimization algorithm used in decision making processes. Its main advantage is the knowledge-free nature [14,15], i.e. the only domain knowledge required is the ability to recognize positive and negative final outcomes of decisions made. The space (problem domain) in which the algorithm operates is represented by a tree structure. The current state is located in the root of a tree. Each node represents particular state and stores information about actions (game moves) taken in this state as well as the respective scores (fractions of simulations that led to a "win" outcome) assigned to that (action, state) pair. Each path in the tree represents a particular sequence of decisions (game moves) with the final outcome being read out in the leaf.

UCT relies on massive random four-stage simulations (see Fig. 1) performed before making a decision:

- **Choice** - starting from the root, go down the tree until a leaf or unexplored node is reached;
- **Expansion** - if possible, create child nodes of previously found node;
- **Simulation/Play-out** - from one of the created nodes, perform a random simulation (game) until the final state;
- **Backpropagation** - populate the result of the above random simulation (game) up the tree, thus update all nodes visited on the path from the root to the leaf node.

The more iterations are performed the better estimations of the true min-max value of each (action, state) pair are obtained, hence the better algorithm's behavior is observed. The issue which needs further explanation is the way nodes are selected in the first stage (Choice). In UCT [13] the strategy of tree expansion is based on the previous simulation outcomes and visit counts. This way the method balances exploration and exploitation factors in order to find the most promising directions of tree growth. In each node X if there exist child nodes (actions) which had not been yet chosen, one of them is selected at random. Otherwise (if all actions had been tried at least once already) the child node k maximizing the following formula is selected:

$$X_j + C\sqrt{\frac{\ln n}{n_j}} \tag{1}$$

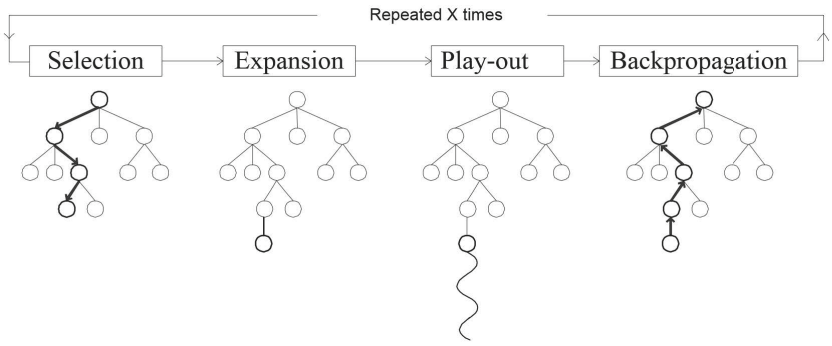

Fig. 1. Operational scheme of the UCT method (reproduced from [2])

where X_j stands for the average score of simulations performed so far from j-th child node, n_j is the number of times the j-th action (child node) was selected while visiting node X, n is the total number of visits to node X and C is the exploration coefficient. In theory C should be equal to $\sqrt{2}$, but experiments show that, for a given problem at hand, the best way is to test multiple values and select the most appropriate one. The most popular applications of UCT are those related to games, in particular the so-called General Game Playing [9,20] and Go [8].

While both VRP and UCT algorithm are commonly known and researched, there are no documented experiments on incorporating UCT into VRP. Thus, to the best of our knowledge, analysis made and described in this paper is very likely to be the first such attempt. Please note that it is the "capacitated" variant of the problem which is considered. Nonetheless, the presented approach may also be suitable for other formulations. In order to adapt UCT into CVRP, several key issues must be considered:

- Representation of the problem in a tree-like structure;
- A method of simulation results evaluation;
- Node selection strategy basing on simulation data.

A tree-like representation is a standard way of describing the current situation in many games, e.g. chess or checkers. As UCT was primary meant for gaming, a corresponding data structure was an obvious choice. However, for CVRP it is not straightforward and natural to translate the possible routes' configurations into tree structure. The first assumption made is that no baseline solution is used and instead routes are built from scratch, i.e. at the beginning all vehicles are located in the depot. This situation is represented by the root node. Going one level down the tree corresponds to assigning one of the clients to a (specific) vehicle. Thus, the height of the tree will be equal to the number of clients. Three methods of making such assignments were analyzed:

- **City-To-Vehicle (CTV)** - each node has k child nodes, where k stands for the number of vehicles. The i-th node corresponds to selection of the i-th vehicle and appending the closest client at end of its current route;

- **Vehicle-To-City (VTC)** - each node has p child nodes, where p stands for the number of unserved clients. The i-th node corresponds to the selection of the i-th unserved client and appending it at the end of the route of the currently closest vehicle;
- **Vehicle-To-City Optimized (VTCO)** - similar to VTC, but unserved client is not automatically assigned to the closest vehicle. Instead, all current routes (i.e. the partial routes of all vehicles) are analyzed in order to find the place minimizing the insertion cost, i.e. the increase of a total routes length due to the insertion of a new client. The client is then added at the specified place.

The first two methods are naive while the third one was created as a result of analysis of their weakness, i.e. proneness to client assignment order caused by the rule of always appending clients at the end of a specified route. VTCO became immune to this issue thanks to the best insertion place search.

Having resolved the first key issue, the next thing was to properly interpret the result of a simulation. While it is trivial in games where (in majority of them) the player can win, lose or draw, in CVRP one obtains a set of routes (one per vehicle) and their total length as a solution. In order for the UCT to expand the tree in the right directions, simulation results must be properly classified as good/poor or promising/unpromising. The problematic issue here is appropriate distinguishing of better routes from the worse ones. Assuming that an approximation of the optimal solution is known (which is the case of the benchmark problems used in this paper), the following formula of solution assessment is proposed:

$$f(x) = \begin{cases} 1 & \text{for } \; x < BEST \\ g(x) & \text{for } \; BEST \leq x \leq 2 \cdot BEST \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where x is the assessed solution's length, $g(x)$ is the inner evaluation function and $BEST$ is the optimal solution length or its approximation. The inner function defines the results' gradation pattern, i.e. the policy of promoting and degrading particular solutions. Three such functions

- **Hyperbolic:**
$$g(x) = \left(\frac{2 \cdot BEST}{x} - 1\right)^2 \tag{3}$$
- **Linear:**
$$g(x) = 2 - \frac{x}{BEST} \tag{4}$$
- **Parabolic:**
$$g(x) = \frac{x}{BEST}\left(2 - \frac{x}{BEST}\right) \tag{5}$$

depicted in Fig. 2 were proposed by the authors. The hyperbolic function (3) concentrates on promoting the very best results while ignoring poor and average ones, the linear function (4) downgrades them proportionally, independent of a

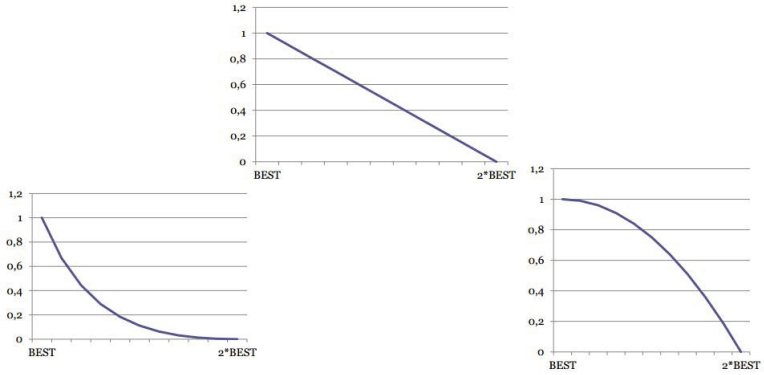

Fig. 2. Inner evaluation functions. From left: hyperbolic, linear and parabolic

distance to the optimal length and the parabolic one (5) has the widest range of solutions which are promoted (treated as promising). As will be seen in the results section the profile of the inner function has a crucial impact on the method's performance.

The last key issue is a suitable decision (node selection) strategy. After a specified number of simulations, a proper choice of the next node (client-vehicle assignment) must be made based on simulation data, i.e. fractions of won simulations in child nodes. Using those one of the child nodes is selected and added to the final solution. It also becomes a root node for the next series of simulations. Three possible strategies were analyzed: **best reward**, **simulations count**, **two-phase method**. The first two strategies are known approaches existing in game domain.

In the best reward strategy, a child with the highest ratio of won simulations to played simulations is chosen. In the simulations count strategy a child with the highest visits count is chosen.

The two-phase method was invented by the authors based on critical analysis of the usefulness of the first two node selection policies in CVRP domain. Please observe that in the game playing suboptimal moves are generally acceptable since the game can often still be won after making such a suboptimal choice. In CVRP, however, assigning a client to a wrong vehicle usually means that the optimal solution is no longer obtainable. In this respect, the best reward strategy or the simulation count policy (both basing on high averages of simulation results) will rather lead to a selection of a subtree containing many good solutions than a subtree containing the best solution and some poor ones. In effect, making wrong decision at the early stage of the algorithm will cost the loss of the optimal result. Two-phase method was designed to partially alleviate this problem. In the first phase child nodes are sorted in descending order based on their average reward values (i.e. won/played ratio). Then, P best children are taken into account in the second phase and among them the child node with the best (indivitwo) solution found during simulations is selected. This way both the most promising regions

Table 1. Test instances. In the instance name X-nY-kZ, X$\in \{A, B, P\}$ denotes type of a benchmark, Y its size, and Z the number of available vehicles

Instance	#Clients	#Vehicles	Capacity	Best solution ($BEST$)
P-n19-k2	19	2	160	212.66
P-n20-k2	20	2	160	220
P-n21-k2	21	2	160	211
P-n22-k2	22	2	160	216
P-n22-k8	22	8	3000	603
P-n23-k8	23	8	40	554
B-n43-k6	43	6	100	747.54
B-n45-k5	45	5	100	751
A-n60-k9	60	9	100	1408
P-n70-k10	70	10	135	834
A-n80-k10	80	10	100	1764
P-n101-k4	101	4	400	681

of a tree (in terms of average results) and the best indivitwo solution found in these regions are taken into account by the node selection policy.

4 Experimental Setup and Results

Experiments were conducted using selected instances of CVRP obtained from [17] whose basic parameters are presented in Table 1. The solution for one of the considered benchmark sets is presented in Fig. 3. The following algorithm parameters' settings were tested:

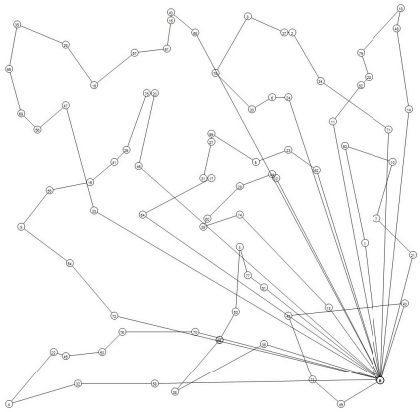

Fig. 3. Solution for the A-n80-k10 benchmark problem

- Assignment method: CTV, VTC, VTCO;
- Inner evaluation function $g(x)$ in (2): hyperbolic, linear, parabolic;
- Decision strategy: best reward, simulations count, two-phase method;
- Exploration factor C in (1);

Table 2. Comparison of top-5 results (in terms of the average results of 50 tests) obtained with CTV, VTC and VTCO methods in test instance with 19 clients and 2 vehicles (P-n19-k2)

Decision	C	AVG	SD	MIN	MAX	Evaluation
City-To-Vehicle						
Simulations	0.5	294.719	21.752	248.764	343.572	Hyperbolic
Simulations	0.2	296.479	23.304	248.764	352.021	Hyperbolic
Reward	1	296.955	22.463	248.764	341.348	Linear
Reward	2	297.347	20.806	248.764	341.348	Linear
Reward	0.2	298.018	22.531	248.764	341.348	Linear
Vehicle-To-City						
Reward	1	252.304	5.313	243.402	261.811	Hyperbolic
Reward	0.2	252.933	4.514	243.402	264.302	Hyperbolic
Reward	0.5	253.459	4.035	243.402	261.811	Hyperbolic
Reward	0.5	253.496	3.360	243.402	263.906	Hyperbolic
Reward	0.2	253.600	3.013	243.402	261.811	Hyperbolic
Vehicle-To-City Optimized						
Reward	2	222.867	4.975	217.964	239.946	Hyperbolic
Reward	1.5	229.535	0	229.535	229.535	Hyperbolic
Reward	0.5	230.018	2.066	226.159	233.636	Hyperbolic
Reward	5	235.741	0	235.741	235.741	Hyperbolic
Reward	15	235.741	0	235.741	235.741	Hyperbolic

– Parameter P in two-phase method: constant value ($P = 5$) or variable selection dependent on the number of clients ($P = 0.25 \cdot$ #Clients).

The number of UCT simulations performed on a single tree level was adaptive and equal to 50 000 on the first (top) level, then was falling linearly down to 5 000 on the penultimate level. Please recall that the number of levels equals the number of clients.

The following data was collected during the experiments: AVG (the average value of results), SD (the standard deviation of results), MIN (the best result found), MAX (the worst result found), EFF (the number of times the MIN result was found). Each experiment consisted of 50 tests based on which the above-mentioned statistics were calculated. Additionally, in the last experiment (summarized in Tables 4 and 5), the ranges of results were analyzed, e.g. 0.5% stands for the number of results falling into the interval $[MIN, MIN(1+0.5\%)]$.

The first set of tests was performed for the smallest (thus computationally inexpensive) benchmark P-n19-k2 and aimed at selection of appropriate ranges or exact settings of the main steering parameters. For each combination of (assignment method, C, inner evaluation function) 50 tests were performed for each of the three main problem mappings: CTV, VTC, VTCO. The values of C belonged to the set $\{0.2, 0.5, 1.0, 1.5, 2.0, 5.0, 15.0\}$. The top-5 results for each mapping are presented in Table 2. The results show that VTCO outperforms both CTV and VTC representations. Not only the average scores are better but also stability, measured by standard deviation is clearly superior. Nevertheless obtained results are far from optimal. This led to analysis and design of two-phase method.

Table 3. Results of two-phase method with VTCO in test instance with 19 clients and 2 vehicles. Column EFF denotes the number of trials in which the MIN value was found in the respective 50 tests.

Decision	C	AVG	SD	MIN	MAX	Evaluation	EFF
Two-phase	0.5	219.845	2.396	212.657	220.643	Hyperbolic	4
Two-phase	1.5	213.793	3.409	212.657	224.02	Hyperbolic	45
Two-phase	5	215.39	4.266	212.657	224.02	Hyperbolic	35
Two-phase	15	214.254	3.194	212.657	220.643	Hyperbolic	41
Two-phase	0.5	218.799	4.333	212.657	226.159	Linear	16
Two-phase	1.5	213.456	2.396	212.657	220.643	Linear	46
Two-phase	5	212.657	0	212.657	212.657	Linear	50
Two-phase	15	212.657	0	212.657	212.657	Linear	50

Based on these initial outcomes we decided to use the two-phase decision strategy and VTCO mapping as well as to skip parabolic inner evaluation function in further tests. We also restricted the tested values of C parameter to the set $\{0.5, 1.5, 5.0, 15.0\}$. The results for each combination of (C, inner evaluation function) for two-phase VTCO are presented in Table 3. Two-phase method proved to be clearly the most efficient among tested approaches for the P-n19-k2 instance. The AVG values are within a few per cent points from the BEST value and in two cases (5, linear) and (15, linear) all 50 tests ended with the $BEST$ score. While VTCO representation and two-phase decision strategy are clearly better than the competitive approaches, the inner evaluation functions do not have such leader. Apart from parabolic function, which gave very poor results, hyperbolic and linear functions have both their better and worse outcomes, however, more data is needed to form any firm conclusion in this matter.

Using the best configuration found, i.e. VTCO representation, two-phase method, hyperbolic and linear evaluation functions and high exploration factors (C=5, 15), the final tests were performed on an ensemble of test instances. The results are presented in Table 4.

The final experiments show that the two-phase VTCO method with proposed configuration performs well on a wide range of test instances. However, there is still room for potential improvement. First of all, it is difficult to tell the influence of P parameter on the test results. In small-size problems a difference between constant value and a calculated one ($P = 0.25 \cdot$#Clients) is minimal. In larger sets no rule can be found as in some tests one option is clearly better while in others it is the other way round. As for the evaluation functions, the situation is similar. For both choices there exist test instances where one is visibly better than the other. Linear function seems to work better for large test instances. Generally speaking, higher exploration factor (C) values provide better results. On the other hand, there are still some cases where the situation is the opposite. Most probably this can be attributed to too large increase of this parameter and therefore some intermediate values of C should be used instead. On a general note, the results support the claim that two-phase method can be regarded as efficient application of the UCT algorithm to solving CVRP. In order to compare

Table 4. Top-3 results of VTCO with two-phase method. Columns denoted $k\%$ present the numbers of results falling into the interval $[MIN, MIN(1 + k\%)]$

Instance	C	P	Evaluation	*AVG*	*SD*	*MIN*	*MAX*	EFF	0.5%	1%	2.5%	5%
P-n19-k2	15	5	Linear	217.367	2.189	212.657	222.387	4	4	4	36	48
	15	4	Linear	217.378	2.735	212.657	222.627	7	7	7	38	48
	5	4	Linear	217.53	2.821	212.657	222.627	6	6	6	39	46
P-n20-k2	5	5	Hyperbolic	217.416	0	217.416	217.416	50	50	50	50	50
	15	5	Hyperbolic	217.416	0	217.416	217.416	50	50	50	50	50
	5	5	Linear	217.469	0.212	217.416	218.309	50	50	50	50	50
P-n21-k2	5	5	Hyperbolic	212.712	0	212.712	212.712	50	50	50	50	50
	5	5	Linear	212.712	0	212.712	212.712	50	50	50	50	50
	15	5	Hyperbolic	212.712	0	212.712	212.712	50	50	50	50	50
	15	5	Linear	212.712	0	212.712	212.712	50	50	50	50	50
P-n22-k2	5	5	Hyperbolic	217.852	0	217.852	217.852	50	50	50	50	50
	5	5	Linear	217.852	0	217.852	217.852	50	50	50	50	50
	15	5	Hyperbolic	217.852	0	217.852	217.852	50	50	50	50	50
	15	5	Linear	217.852	0	217.852	217.852	50	50	50	50	50
P-n22-k8	5	5	Linear	605.833	4.681	601.424	616.633	7	30	34	49	50
	5	5	Hyperbolic	606.441	4.743	601.424	616.639	7	23	31	48	50
	15	5	Linear	611.746	7.282	601.424	628.255	5	12	15	35	50
P-n23-k8	5	5	Hyperbolic	534.665	5.033	531.174	555.088	17	33	38	47	50
	5	5	Linear	535.522	5.540	531.174	558.924	12	29	38	46	49
	15	5	Hyperbolic	536.290	7.687	531.174	575.287	5	21	39	47	48
B-n43-k6	15	10	Linear	771.621	9.230	760.211	803.806	1	14	21	42	49
	5	10	Linear	771.624	7.843	756.949	790.680	1	4	9	39	50
	5	10	Hyperbolic	771.903	7.320	747.891	790.981	1	1	1	12	49
B-n45-k5	5	5	Hyperbolic	773.888	6.690	760.315	790.599	1	4	9	38	50
	5	11	Hyperbolic	773.967	6.287	756.580	797.029	1	1	1	32	49
	5	11	Linear	775.486	6.262	763.029	790.939	1	4	11	42	50
A-n60-k9	15	15	Hyperbolic	1460.932	30.257	1390.822	1533.801	1	2	3	4	27
	5	15	Linear	1461.015	30.044	1406.876	1524.494	1	2	7	13	34
	15	15	Linear	1465.919	27.417	1402.179	1526.206	1	1	2	8	29
P-n70-k10	5	17	Linear	929.452	18.142	879.660	965.643	1	1	1	5	17
	5	17	Hyperbolic	933.520	17.375	898.108	969.764	1	2	3	11	37
	5	5	Linear	933.928	22.629	889.544	990.779	1	2	3	9	27
A-n80-k10	5	20	Hyperbolic	1940.641	30.098	1889.752	2009.415	1	5	9	26	45
	15	20	Hyperbolic	1941.046	28.495	1883.619	2013.686	1	2	2	22	45
	5	20	Linear	1943.568	35.479	1882.487	2012.718	1	3	5	22	42
P-n101-k4	5	5	Hyperbolic	730.058	6.899	712.358	748.911	1	1	2	26	49
	5	5	Linear	730.617	7.238	714.500	747.140	1	2	6	32	50
	15	5	Hyperbolic	732.697	6.322	718.433	748.421	1	2	7	35	50

the results with an external approach adequate tests with a simplified version of the 2-phase PSO method [18,19] were performed on same test instances. The results are presented in Table 5.

A comparison of results presented in Tables 4 and 5 shows that the overall results of 2PSO are a few percent points better than those of UCT, especially for larger problem instances. It should be noted, however, that 2PSO is a highly complex optimization approach which uses both PSO and 2-opt local optimization. Moreover, the differences are relatively small, even though only the baseline UCT implementation was tested in our approach, which did not include any enhancements, commonly used in games domain. Hence, we believe that proposed approach has potential which we plan to continue investigating in our future research.

Table 5. Test results of a simplified version of a 2-phase PSO approach

Test	*AVG*	*MIN*	*MAX*	*EFF*	0.5%	1%	2.5%	5%
P-n19-k2	213.13	212.66	226.02	45	45	45	50	50
P-n20-k2	219.94	219.94	219.94	50	50	50	50	50
P-n21-k2	213.26	212.71	218.31	46	46	50	50	50
P-n22-k2	220.05	217.85	225.68	38	38	49	49	50
P-n22-k8	607.01	600.83	742.12	32	34	46	46	50
P-n23-k8	531.17	531.17	531.17	50	50	50	50	50
B-n43-k6	757.82	746.98	871.31	1	15	29	48	50
B-n45-k5	766.72	754.22	921.59	3	13	20	43	50
A-n60-k9	1415.51	1374.83	1700.44	1	3	11	42	49
P-n70-k10	896.18	846.66	1132.65	1	1	2	18	39
A-n80-k10	1888.69	1796.51	2272.38	1	1	4	13	38
P-n101-k4	719.25	706.19	827.07	1	4	18	46	50

5 Conclusions and Future Work

In this paper, a novel approach to solving the NP-Hard CVRP based on the UCT method was proposed and experimentally evaluated. In order to adapt the UCT formulation to this new problem domain a two-phase node selection procedure, which breaks the classical UCT selection scheme, was proposed and experimentally verified. With larger exploration factors and appropriate choice of the internal evaluation function the results are very promising and only slightly inferior to those accomplished by a complex two-phase PSO algorithm.

In the future we plan to verify the efficiency of several UCT modifications (commonly used in games) in CVRP domain, e.g. Rapid Action Value Estimation [8] or weighted simulations [21]. We believe that application of the enhancements which proved to be efficient in games domain may lead to further improvement of the CVRP results and strengthen the claim about potential applicability of the UCT method beyond games.

Acknowledgements. The research was financed by the National Science Centre in Poland, based on the decision DEC-2012/07/B/ST6/01527.

References

1. Breedam, A.V.: 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. Ph.D. thesis, University of Antwerp, Belgium (1994)
2. Chaslot, G., Winands, M.H.M., Szita, I., van den Herik, H.J.: Cross-Entropy for Monte-Carlo Tree Search. ICGA Journal (3), 145–156 (2008)
3. Clarke, G., Wright, J.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12(4), 568–581 (1964)
4. Dantzig, G.B., Ramser, J.: The truck dispatching problem. Management Science 6(1), 80–91 (1959)
5. Dorigo, M.: Optimization, Learning and Natural Algorithms. Ph.D. thesis, Politecnico di Milano (1992)
6. Eilon, S., Watson-Gandy, C., Christofides, N.: Distribution Management: Mathematical Modelling and Practical Analysis, 1st edn., Griffin (January 1976)

7. Fisher, M., Jaikumar, R.: 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)
8. Gelly, S., Silver, D.: Monte-carlo tree search and rapid action value estimation in computer go. Artificial Intelligence 175(11), 1856–1875 (2011)
9. Genesereth, M.R., Love, N., Pell, B.: General game playing: Overview of the aaai competition. AI Magazine 26(2), 62–72 (2005)
10. Gillett, B., Miller, L.: A heuristic algorithm for the vehicle dispatch problem. Operations Research 22(2), 340–349 (1974)
11. Lenstra, J.K., Rinnooy Kan, A.R.K.: Complexity of vehicle routing and scheduling problems. Networks 11, 221–227 (1981)
12. Khouadjia, M.R., Alba, E., Jourdan, L., Talbi, E.-G.: Multi-Swarm Optimization for Dynamic Combinatorial Problems: A Case Study on Dynamic Vehicle Routing Problem. In: Dorigo, M., et al. (eds.) ANTS 2010. LNCS, vol. 6234, pp. 227–238. Springer, Heidelberg (2010)
13. Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 282–293. Springer, Heidelberg (2006)
14. Mańdziuk, J.: Knowledge-Free and Learning-Based Methods in Intelligenet Game Playing. SCI, vol. 276. Springer, Heidelberg (2010)
15. Mańdziuk, J.: Towards cognitively-plausible game playing systems. IEEE Computational Intelligence Magazine 6(2), 38–51 (2011)
16. Christofides, N., Mingozz, A., Exact, P.T.: algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. Mathematical Programming 20(1), 255–282 (1981)
17. Networking, N.: Emerging Optmization (2013), http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/
18. Okulewicz, M., Mańdziuk, J.: Application of Particle Swarm Optimization Algorithm to Dynamic Vehicle Routing Problem. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2013, Part II. LNCS(LNAI), vol. 7895, pp. 547–558. Springer, Heidelberg (2013)
19. Okulewicz, M., Mańdziuk, J.: Application of Particle Swarm Optimization Algorithm to Dynamic Vehicle Routing Problem. In: Proceedings of the 2nd IEEE Symposium on Computational Intelligence for Human-Like Intelligence, pp. 86–93. IEEE Press (2014)
20. Świechowski, M., Mańdziuk, J.: Self-adaptation of playing strategies in general game playing. IEEE Transactions on Computational Intelligence and AI in Games 6(4), 367–381 (2014)
21. Xie, F., Liur, Z.: Backpropagation modification in monte-carlo game tree search. In: IITA 2009 Proceedings of the 2009 Third International Symposium on Intelligent Information Technology Application, vol. 2, pp. 125–128 (2009)