A Strategic Planning Architecture and Computational Tournament for the Competitive Prize Collection Problem

Mark R. Johnston
Landcare Research
Palmerston North
New Zealand
JohnstonM@landcare.cri.nz

John W. Giffin
Massey University
Palmerston North
New Zealand
J.W.Giffin@massey.ac.nz

Abstract

The Competitive Prize Collection Problem (CPCP) involves two autonomous players competing to collect a fixed number of contestable prizes located in the Euclidean plane. We propose a Strategic Planning Architecture (SPA) for hierarchically structuring the decision problem of each player in terms of planning horizon, aggregation of prizes, contingency planning, cognizance of opponent, and dynamic response monitoring. To evaluate the difficulty of problem instances, and compare the performance of player strategies, we design a computational tournament. However, the effectiveness of a player strategy for the CPCP cannot be evaluated in isolation from other strategies, since a strategy can only be evaluated against another strategy, one for each player. In addition, to address the circular nature of benchmark problem instances and effective strategies, we consider first robustness of performance on a general class of problem instances followed by expected performance on a class of bad-case problem instances.

1 Introduction

Johnston and Giffin [3, 4, 5, 6] introduced the **Competitive Prize Collection Problem** (CPCP). Let $V = \{1, \ldots, n\}$ be a set of "contestable" *prizes*. Associated with each prize $i \in V$ is a *location* (x_i, y_i) in the Euclidean plane and a *value* $v_i > 0$. For the CPCP, two independent *players*, $\mathcal{A}$ and $\mathcal{B}$, with given initial locations, (x_A, y_A) and (x_B, y_B) respectively, move continuously at the same constant speed in the Euclidean plane. Each player has the same objective: to collect as much in total prize value as possible until the overall deadline, λ, expires. The value of each prize is only awarded to the first player who visits. A prize location that is visited simultaneously by both players has its associated value shared equally. At all times, each player has perfect observation of the state of the game position, i.e. where the players are currently located and which prizes remain unclaimed. Fekete and Schmitt [1] study a similar problem called the

Competing Salesmen Problem (CSP), in which players take turns, moving one edge at a time within a graph, with customers located at the vertices of the graph. They focus on establishing whether a player can avoid a loss or force a draw in the graph-based CSP, mainly considering the special case where the graph is a tree.

A *game* is a description of strategic interaction involving two or more decision makers called players. It includes constraints on actions that players can take and players' interests, but does not specify the actions that players *do* take (Osborne and Rubinstein [8]). The CPCP is a *perfect information* game, i.e. each player has complete information about the opponent's position and about the choices available to the opponent. When there is only a single player, the corresponding decision problem is a well-defined optimization problem, which may be computationally difficult to solve but only the choices of the single decision maker determine the final outcome. When there are multiple players, no one player completely controls the final outcome, so what is meant by a good decision must be defined before an attempt is made to find one. Computationally, a strategy is an algorithm that, given the current state of the game, will determine exactly one legal movement. In the CPCP, players determine their actions simultaneously, observe the new state of the game following the execution of that action, and repeat the procedure until the game terminates. Each player's decisions are dependent upon the current location, the opponent's location and previous actions, and the location and value of the remaining prizes. The difficulty is how to use the limited computational resource effectively.

This paper is concerned with two major engineering cornerstones in the study of the CPCP. The first (Section 2) is the design of a generic strategy architecture for hierarchically structuring the decision problem of each player, within which a more specific strategy may be developed. Although any number of strategies may be proposed for the CPCP, we identify those structures and features common to a number of strategies. The second (Section 3) is the design of computational experiments to evaluate the effectiveness of strategies and account for differences in difficulty of problem instances. Establishing benchmark sets of both problems instances and opposing strategies raises a number of significant issues that do not arise in the corresponding evaluation of heuristics for combinatorial optimization problems. Section 4 draws some conclusions and suggests some future research directions.

2 Strategic Planning Architecture

Strategic planning necessarily involves both long-term, medium-term and short-term considerations. A *scenario* is a description of the future and the process of how to get there from the present (Meristö [7]). Within a planning horizon, a number of scenarios regarding the movements of both players and the prizes they collect may be proposed. In order to evaluate and compare these scenarios, some assumptions about the *rationality* (consistency in pursuit of objectives) and *intelligence* (knowing everything we can infer about the situation) of the opponent are required. Once the future consequences of various scenarios are understood, it then remains to decide upon some action in the immediate-term. These three building blocks — the interaction of planning horizons, contingent planning and dynamic response to the opponent's behaviour — form the generic architecture that is developed in this section.

2.1 Hierarchical Decision Structure

The necessity to make decisions implies that there is a context in which a decision problem is formulated. This context, which we will call a *frame*, describes the prizes that are relevant and accessible, the structural aggregation of the prizes, a player, and the opponent. Suppose we fix a planning horizon and restrict the regional focus on prizes considered. The basic decision problem is to determine how the game is likely to evolve over the duration of this planning horizon. The players may also have directives to follow or surrogate objectives. The following attributes are sufficient to define a frame, and are defined as follows:

Horizon. The planning horizon represents how far into the future to plan, and can be *fixed, variable, rolling or dynamic*.

Scope. The scope describes the subset of prizes that may be considered, how the visibility of the players may be limited, how the prizes may be aggregated, the structure of the aggregation (e.g., into clusters), and any stratification of the prizes.

Directives. A directive is a constraint placed upon the decision problem. These constraints may include restricted displacement of the player, target locations, time windows on harvesting within favourable regions, maintaining an avoidance distance from the opponent or some lower bound on harvesting rate.

Surrogates. Surrogate objectives describe the goal or purpose of the decision problem as these may be different from the overall objective of the CPCP. These may include harvesting rate, time limit to reach a target location, or milestones to be attempted.

The **Strategic Planning Architecture** (SPA) is a hierarchy of frames. The *global-frame* is the initial decision problem considered in a strategy, and has infinite planning horizon, full scope (i.e. all prizes are available), no directives and no surrogates. The global-frame selector not only selects a strategic plan but the implementation of that plan is itself a frame spawned by the selector. Hence, the decision made by a selector process is another frame. At each frame, the selector defines a decision problem whose solution is a sub-frame of a finer level, i.e. the scope is smaller (there are fewer available prizes), the planning horizon is shorter, and more specific directives and surrogates are applied. The coarseness of the sub-frame could be different, depending on which decision was made. Hierarchical structure provides for the decoupling of decisions at different planning horizons into strategic, tactical and operational planning. Moreover, the decisions telescope since we make progressively finer decisions, at decreasing levels of aggregation, until an actual *step* is determined.

The aim of aggregation is to isolate a natural strategic structure, local to a particular frame, suitable and sufficient to determine how the game may evolve over the planning horizon of that frame. The number of frames used, their relative planning horizons and their planning scopes are dynamic and should adapt to the natural structure of the evolving game state. However, this depends very much on what structure (or lack thereof) there is to exploit. Hence, it may be necessary to impose some structure when it is necessary to decompose the decision problem but no natural structure is apparent. We

wish to maintain a balance between planning horizons but know there is always a trade-off between scope and detail concomitant with a computational budget. Although the framework is sufficiently flexible to make frame choice and goal-within-frame choice independent, there is a synergy between the choice of frames and the methods used within the adjoining frames. A method at one frame attempts to exploit the structure of the objects within that frame and, hence, any subframe will be based around the object selected. In addition, there is strong coupling between frames, since decisions in one frame implement and refine the decisions of the previous frames and model the focussed prize region in more detail.

2.2 Scenarios and Contingent Evaluation

A scenario need not be an exact future course but rather an approximate representation. Multiple scenario analysis involves the development of a representative set of mutually exclusive alternatives and assessing the player's response to each of these possible futures.

Contingency is the analysis of the consequences of a set of actions incorporating conditioning of the future plan upon some classification of future state(s), i.e. **if** encounter future *state1* **then** take *action1* **else if** encounter future *state2* **then** take *action2*. Ideally, we would like to be able to plan for every possible future decision of our opponent. However, given a scarce computational resource, this is clearly unrealistic as contingency planning is computationally expensive. Therefore, contingency planning must be balanced against both *aggregation* (the process of selecting and organising significant groups of prizes) and the *certainty* of the evaluation of each scenario. Thus, the issue becomes how to make effective use of the computational resource available to solve a range of problem sizes and structures. In other word, we consider both the "forest" and the "trees" simultaneously.

Séguin, Potvin, Gendreau, Crainic and Marcotte [9] outline the functional components of a real-time decision process:

(i) Information management and data fusion.

(ii) Situation assessment and evaluation of alternatives.

(iii) Decision: a decision does not necessarily result in an immediate action, rather, the action is incorporated into a plan that extends over a rolling time period known as a planning horizon.

These functional components are implemented as two processes that operate within the context of a frame. The **planner** generates and evaluates at least one scenario, but need not involve contingent planning. This constitutes components (i)–(ii) above. For example, NEAREST NEIGHBOUR could be used as a single scenario planner, or selecting random prizes could be used as a multiple scenario planner. The **selector** decides upon a scenario from those generated by the planner and implements the initial actions of the plan. This constitutes (iii) above. Even if only one scenario is proposed by the planner, the selector may be non-trivial as it must determine how much of the initial plan to implement.

2.3 Dynamic Response Monitoring

Séguin, Potvin, Gendreau, Crainic and Marcotte [9] distinguish between three types of planning in a real-time decision process:

Reactive planning. Short-term, local effect, small changes in global state.

Incremental planning. Modification, global state does not depart too much from expected state at the time the plan was first devised.

Deliberative planning. Mandatory revision when state of the world departs significantly from its predicted state.

While contingent planning attempts to account for the future actions of the opponent, observation of the opponent is necessary to determine whether reactive, incremental or deliberative planning is required at each frame within the SPA. **Response Monitoring** is a cycle of forecasting the likely targets of the opponent, observation of whether the opponent's behaviour is consistent with our prediction, plan refinement and, when necessary, triggering of deliberative planning. Each player must decide how much cognizance they take of their opponent's movements and opportunities.

A **Dynamic Monitoring System** (DMS) is a particular implementation of the SPA that applies a *scenario engine* as its *planner* module and a *monitor* as its *selector* module:

Scenario Engine. Recommends the action that should be taken for each possible target set of the opponent, proposes and evaluates a number of scenarios using a contingent evaluator.

Monitor. Determines the target set that the opponent is currently targeting or likely to target and spawns the sub-frame corresponding to that scenario.

Monitoring consists of forecasting, observation and scenario selection. A **forecast** assumes that it is possible to predict the future, assigning probabilities to the occurrence of each possible future. Thus, a forecast is an estimate of what is likely to occur. An **observation** is the noting of an event as it occurs.

We have developed an implementation of the SPA/DMS consisting of four dynamic frames (see Johnston [4] for details). The coarsest frame, the *grid-frame,* considers the density of prize value. The *cluster-frame* considers contingent sequences of clusters, the *prize-frame* considers contingent sequences of prizes, and the finest frame, the *step-frame,* considers individual steps towards a single prize or a pair of prizes.

3 Computational Tournaments

Two significant CPCP research questions can be addressed via computational experimentation only:

Understanding Problem Instances. We wish to determine what makes a problem instance difficult, and how to design difficult test problem instances. This involves

defining a number of problem classes, generating a large number of random problem instances from each class, determining which classes are seemingly more difficult, and designing a number of *bad-case* problem instances from each of the most difficult problem classes.

Effectiveness of Strategies. We wish to understand what a strategy needs to address to be successful and which strategies are effective on various problem classes. We can identify which are the most promising strategies in terms of worst performance, expected performance, and their nemesis.

Computational experimentation with vehicle routing and scheduling problems usually compares the relative quality of solutions of different heuristics in solving specific classes of problems. However, the effectiveness of a strategy for the CPCP cannot be evaluated in isolation from other strategies, since a strategy can only be evaluated against another strategy, one for each player.

Computational experiments that compare the performance of strategies take the form of a *computational tournament* in which a number of strategies play off against one another on a set of problem instances. A **problem instance**, $\wp$, is completely defined by the number of prizes, the location of each prize, the value of each prize, the initial location of each player, the overall deadline, and the step size, Δ. A **simulation battle** is a single simulated play of the CPCP on a particular problem instance with a particular pair of strategies, one for each player. This constitutes a (simulated) dynamic evaluation of the chosen strategies on the given problem instance. There are, however, two principal difficulties in evaluating the relative effectiveness of strategies:

(i) How should the performance of a particular strategy against a number of opposing strategies be weighted when some strategies may be more difficult to play against than others.

(ii) How should the performance between two strategies, over a number of problem instances, be weighted when some problem instances may be more difficult than others.

Hence, the effectiveness of strategies and the difficulty of problem instances are not independent investigations. Problem instances are required to evaluate the performance of strategies, and strategies are required to evaluate the difficulty of problem instances. To measure a strategy's success we must consider both robustness (worst-case performance) and expected performance (average-case performance). Moreover, Hooker [2] argues that the first heuristics designed for a new problem determine the benchmark problem set against which future heuristics will be evaluated — since these are the problem instances the heuristic does well on — and, circularly, the benchmark problem set determines the future heuristics designed for the problem — since they must perform well on the benchmark problems to be considered good.

To address all these issues we adopt the following four-step approach:

Step I. Specification of general classes of problem instances (Section 3.2).

Step II. Preliminary tournaments between a range of strategies, on the general problem instance classes. Evaluate which strategies are most robust on each problem class by worst-case performance on average-case problems (Section 3.3).

Step III. Static estimation of the expected value of a problem instance for each player (Section 3.1), and sensitivity analysis of problem instances, to design bad-case problem instances (Section 3.2).

Step IV. Final tournaments between the most robust strategies of the preliminary tournaments on a set of bad-case problem instances from each of the most difficult problem classes. Evaluation is on the basis of expected, average-case performance on bad-case problem instances (Section 3.3).

3.1 Expected Value of the Game

A strategy will often wish to estimate the value $v_A^*(\wp)$ or $v_B^*(\wp)$ of a problem instance, usually under some scenario of how the immediate future will proceed. A **dynamic estimator** of the value of a problem instance is a procedure that estimates $v_A^*(\wp)$ from the results of at least one simulation battle. A **static estimator** of the value of a problem instance is a procedure that estimates $v_A^*(\wp)$ without performing any simulation battles.

Two-person game theory is divided into **constant sum games**, in which the sum of the payoffs is constant over every pair of player actions, and **general sum games**, in which the sum of the payoffs need not be constant over pairs of player actions (Osborne and Rubinstein [8]). Competition is **perfect** in a constant sum game since any payoff which one player does not receive must be received by the other player. However, in a general sum game, competition is not perfect. General sum games are further subdivided into **noncooperative**, in which any type of collusion, such as correlated strategies and side payments, is forbidden, and **cooperative**, in which all such cooperation is permitted. A noncooperative game focuses on sets of possible actions of individual players, whereas a cooperative game (or **coalition game**) focuses on the sets of possible joint actions of groups of players. When $\lambda = \infty$ the CPCP is a constant sum game since there is always time to claim one more prize. However, when $\lambda < \infty$ the CPCP is a noncooperative general-sum game, since it is possible that the overall deadline expires before all prizes are claimed. In the latter case, the players may exhibit qualities of cooperation, since both players could be better off by not entering into time consuming conflict.

Let $v_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$ denote the total prize value claimed by player $\mathcal{A}$ in a simulation battle on problem instance $\wp$ in which player $\mathcal{A}$ adopts strategy a and player $\mathcal{B}$ adopts strategy b. Let $\mathbb{S}_\infty$ be the infinite set of all possible player strategies. The **value of the game** $\wp$ (or the **value of problem instance** $\wp$), $v_A^*(\wp)$, is defined as the *Nash equilibrium value* in mixed strategies (see Osborne and Rubinstein [8]) to player $\mathcal{A}$ of the infinite two player game in which both players' pure strategies are $\mathbb{S}_\infty$, if such a Nash equilibrium exists. Let $\mathbb{S}$ be a given finite set of player strategies. The **computational maximin value of the game** $\wp$ (or the **computational maximin value of the problem instance** $\wp$), $v_A^\diamond(\wp)$, for the given $\mathbb{S}$, is

$$v_A^\diamond(\wp) = \max_{\mathsf{a} \in \mathbb{S}} \left\{ \min_{\mathsf{b} \in \mathbb{S}} v_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) \right\}$$

Finally, $\upsilon_B(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$, $\upsilon_B^*(\wp)$ and $\upsilon_B^\diamond(\wp)$ are defined similarly for player $\mathcal{B}$.

3.2 Classes of Problem Instances

The **prize class** (P-class) focuses on the layout and value of individual prizes, generated by construction, or drawn from some probability distribution, or through a combination of these. The **cluster class** (C-class) focuses on the layout and value of individual clusters of prizes. The **density class** (D-class) focuses on the composition of prize value density features. These problem instance classes can be described as *average-case* problems since there is no prior reason to expect any one problem instance to be any more difficult than any other.

A problem is strategically difficult either when experimentation shows that payoff estimated by tactical or strategic analysis is not realised (predictability), or when small variations in the problem instances produce significant changes in the tactics required to realise the expected payoff (sensitivity). We expect that an *easy problem instance* would correspond to one for which it is easy to estimate its value, i.e. the better static estimators would give approximately the same value. Hence, one way to define a *difficult problem instance* is as a problem instance for which there is a high variability between estimators of the value of that problem instance, i.e. a difficult problem instance is one that is hard to predict accurately and hence hard to make good strategic and tactical decisions for.

To define a quantitative measure of problem instance difficulty, choose two (static or dynamic) estimators, υ' and υ''. Define the difficulty, $\xi(\wp)$, of problem instance $\wp$ by

$$\xi(\wp) = \frac{(|\upsilon'_A(\wp) - \upsilon''_A(\wp)| + |\upsilon'_B(\wp) - \upsilon''_B(\wp)|)}{2\Omega(\wp)}$$

where $\Omega(\wp)$ is the maximum possible value in prizes that the two players can jointly collect if they completely cooperate. A problem instance $\wp$ is, therefore, *bad-case* if $\xi(\wp)$ is significantly nonzero.

The *construction and improvement* paradigm may be applied to finding bad-case problem instances. "Improving" the difficulty of an existing constructed problem instance involves perturbing the initial player locations, prize locations, prize values, or overall deadline. We propose that bad-case problem instances can be found by searching possible initial player locations for player $\mathcal{A}$ and player $\mathcal{B}$ initial locations that maximize $\xi(\wp)$, while fixing the prize locations, prize values, and the overall deadline. Local search is not appropriate since the discrete nature of $\xi(\wp)$ may lead to a spatial plateau of constant values, with respect to perturbations in one spatial variable, which (non-metaheuristic) local search often finds difficult to search successfully.

3.3 Effectiveness and Robustness of Strategies

We can evaluate $\upsilon_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$ and $\upsilon_B(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$ by dynamic simulation battle. However, one player may be able to accumulate more value than the other simply because of the inequity of the starting locations. A solution to this problem is to repeat the simulation battle with the initial player locations reversed and let the score be the sum of the original and reversed battle results. We expect that strategies of approximately

equivalent effectiveness will claim approximately the same total in prizes from the two simulation battles. We also expect that if one strategy dominates another then it will do so, on average, from both orientations of the initial player locations.

A further issue is the comparability of a resulting score on one problem instance with the score on another. Benchmarking the score against the expected value of the problem instance would be ideal, but we must settle for the computational maximin value of the problem instance. In addition, the total prize pool for each problem instance is standardised. Hence, the resulting scores, $\varkappa_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$ for player $\mathcal{A}$ and $\varkappa_B(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$ for player $\mathcal{B}$, for the simulation battle between player $\mathcal{A}$ adopting strategy a and player $\mathcal{B}$ adopting strategy b on problem instance $\wp$, are defined by

$$\begin{aligned}
\varkappa_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) &= \upsilon_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) - \upsilon_A^{\diamond}(\wp) \quad + \\
&\quad \upsilon_B(\wp; \mathcal{A} \sim \mathsf{b}, \mathcal{B} \sim \mathsf{a}) - \upsilon_B^{\diamond}(\wp) \\
\varkappa_B(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) &= \upsilon_B(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) - \upsilon_B^{\diamond}(\wp) \quad + \\
&\quad \upsilon_A(\wp; \mathcal{A} \sim \mathsf{b}, \mathcal{B} \sim \mathsf{a}) - \upsilon_A^{\diamond}(\wp)
\end{aligned}$$

Consider the maximin score, $\varkappa^{\diamond}(\wp)$, over a problem instance $\wp$:

$$\begin{aligned}
\varkappa^{\diamond}(\wp) &= \max_{\mathsf{a}\in\mathbb{S}} \min_{\mathsf{b}\in\mathbb{S}} \varkappa_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) = \max_{\mathsf{b}\in\mathbb{S}} \min_{\mathsf{a}\in\mathbb{S}} \varkappa_B(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) \\
&= \max_{\mathsf{a}\in\mathbb{S}} \min_{\mathsf{b}\in\mathbb{S}} \left(\upsilon_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) + \upsilon_B(\wp; \mathcal{A} \sim \mathsf{b}, \mathcal{B} \sim \mathsf{a})\right) \\
&\quad - \left(\upsilon_A^{\diamond}(\wp) + \upsilon_B^{\diamond}(\wp)\right).
\end{aligned}$$

That is, $\varkappa^{\diamond}(\wp)$ is the difference between the *maximin of the sum* of the total prize value from each orientation of the initial player locations and the *sum of the maximin* of the total prize value from each orientation separately. If an *optimal* (Nash equilibrium) strategy were included in $\mathbb{S}$ then $\varkappa^{\diamond}(\wp) = 0$. This implies that $\varkappa_A$ and $\varkappa_B$ are not biased on any class of problem instances.

It remains to define the robustness and effectiveness of an individual strategy. This can only be evaluated for to a set of opposing strategies, $\mathbb{S}$, and a set of problem instances, $\mathbb{P}$. However, the evaluation is different for the preliminary and final tournaments. For the preliminary tournaments, we wish to evaluate the robustness of a strategy, i.e. determine the worst-case result against an average-case set of problem instances and opponent strategies. Hence, we define the robustness of strategy a as the 'MIN-MIN' evaluation

$$\varkappa_A(\mathbb{P}; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathbb{S}) = \min_{\mathsf{b}\in\mathbb{S}} \min_{\wp\in\mathbb{P}} \varkappa_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}).$$

For the final tournaments, we wish to evaluate the expected performance of a strategy, i.e. determine the average-case result against a bad-case set of problem instances and opponent strategies. Hence, we define the effectiveness of strategy a as the 'MEAN-MEAN' evaluation

$$\varkappa_A(\mathbb{P}; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathbb{S}) = \frac{1}{|\mathbb{S}|} \sum_{\mathsf{b}\in\mathbb{S}} \frac{1}{|\mathbb{P}|} \sum_{\wp\in\mathbb{P}} \varkappa_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}).$$

4 Conclusions

This paper has outlined two important contributions to the study of the CPCP. The strategic planning architecture gives some general structure to the decision problem faced by a player in designing a strategy. Contingent analysis of multiple scenarios is used in the planning phase, and forecasting and observation are used in the selection phase. The experimental design of the computational tournament — as a means of evaluating the effectiveness of strategies and the difficulty of problem instances — also raised a number of difficulties in terms of benchmarking against sets of opposing strategies and classes of problem instances, fairness of the initial player locations, and the comparability of a numerical score across problem instances. Although no specific strategies were defined, nor any tournament results presented, these design issues illustrate the significant difference between the CPCP and standard vehicle routing and scheduling problems.

References

[1] FEKETE, S., AND SCHMITT, M. Traveling salesmen in the age of competition. ZPR 97.266, Center for Parallel Computing, Universität zu Köln, 1997.

[2] HOOKER, J. N. Testing heuristics: we have it all wrong. *Journal of Heuristics 1* (1995), 33–42.

[3] JOHNSTON, M. R. A game tree search-based heuristic strategy for the competitive prize collection problem. In *Proceedings of the 32nd Annual ORSNZ Conference* (1996), Operational Research Society of New Zealand, pp. 137–142.

[4] JOHNSTON, M. R. *Dynamic Routing with Competition: Foundations and Strategies*. PhD thesis, Massey University, 1999.

[5] JOHNSTON, M. R., AND GIFFIN, J. W. Modelling a competitive prize-collecting TSP. In *Proceedings of the 30th Annual ORSNZ and 45th Annual NZSA Conference* (1994), Operational Research Society of New Zealand, pp. 172–177.

[6] JOHNSTON, M. R., AND GIFFIN, J. W. Exploring competitive prize collection. In *Proceedings of the 31st Annual ORSNZ Conference* (1995), Operational Research Society of New Zealand, pp. 5–12.

[7] MERISTÖ, T. Not forecasts but multiple scenarios when coping with uncertainties in the competitive environment. *European Journal of Operational Research 38* (1989), 350–357.

[8] OSBORNE, M. J., AND RUBINSTEIN, A. *A Course in Game Theory*. MIT Press, Cambridge, Mass., 1994.

[9] SÉGUIN, R., POTVIN, J.-Y., GENDREAU, M., CRAINIC, T. G., AND MARCOTTE, P. Real-time decision problems: an Operational Research perspective. *Journal of the Operational Research Society 48* (1997), 162–174.